_IncludeLibrary

_LearnCocos2DBookThumbnail

_NotYetReleasedDisclaimer

This Kobold2D version is not available yet!
This page describes changes to Kobold2D which have been implemented, but the Kobold2D version it refers to has not been released yet.

_scrap migrating cocos2d project

Make a backup of your project just in case!

This is a general guide showing you the principles of converting an existing Cocos2D project to Kobold2D. It does not offer a solution to every possibly required change to successfully migrate a project. Each project is different and may require custom modifications not outlined in this guide.

1. create a kobold2d project from Hello Kobold2D template project
    - remove unneeded files:
        a. remove HelloWorldScene class (.h/.m)
        b. remove platform target (iOS or Mac) that isn't supported by your project
        c. remove platform specific scheme (iOS or Mac) that isn't supported by your project
        d. remove the platform-specific Projectfiles-XXX group that isn't supported by your project
        e. remove unneeded Resource files:
            - ship.png
            - ship-hd.png
            - Pow.caf
2. modify config.lua:
    - change FirstSceneClassName to your project first scene's name

        In that scene the class method "+(id) scene" is not called. Any required code in that class needs to be moved to the scene's init method.

    - change any other settings of your project deviating from the default settings, for example Orientation, Color format, Pixel format, Retina support, etc.

        This is where most issues will arise when migrating. Be sure to check all the config.lua settings for changes regarding cocos2d startup.

3. Add Files to "project name"
    a. select all source code files, except:
        - AppDelegate
        - RootViewController
        - GameConfig.h

        - any other library source code that is already provided by Kobold2D (eg. Cocos2D, etc.)
    b. make sure to check "copy to destination"
4. Add Files to "project name"
    a. select all of your project's unique resources, be sure not to include:
        - Info.plist
        - fps_images.png
        - any Cocos2D resources that you did not modify, for example: Icon.png, Default.png, ...
    b. make sure to check "copy to destination"

_Template-Images

_Doodle-Drop-Template_Images

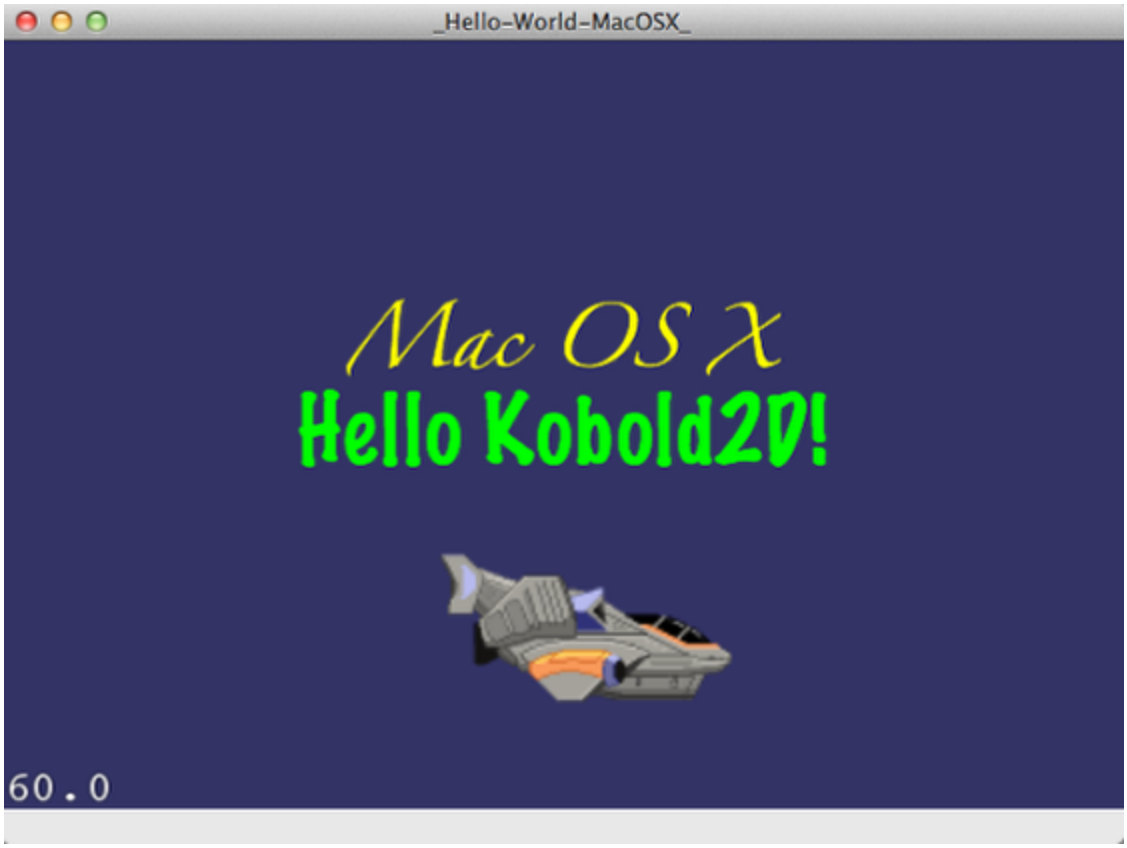

11
GAME OVER!

_Hello-Cocos3D-Template_Images


Hello Kobold2D!
60.0

_Hello-Kobold2D-Template_Images


Test Advertisement
iAd
iPhone/iPod Touch Simulator
Hello Kobold2D!
59.9



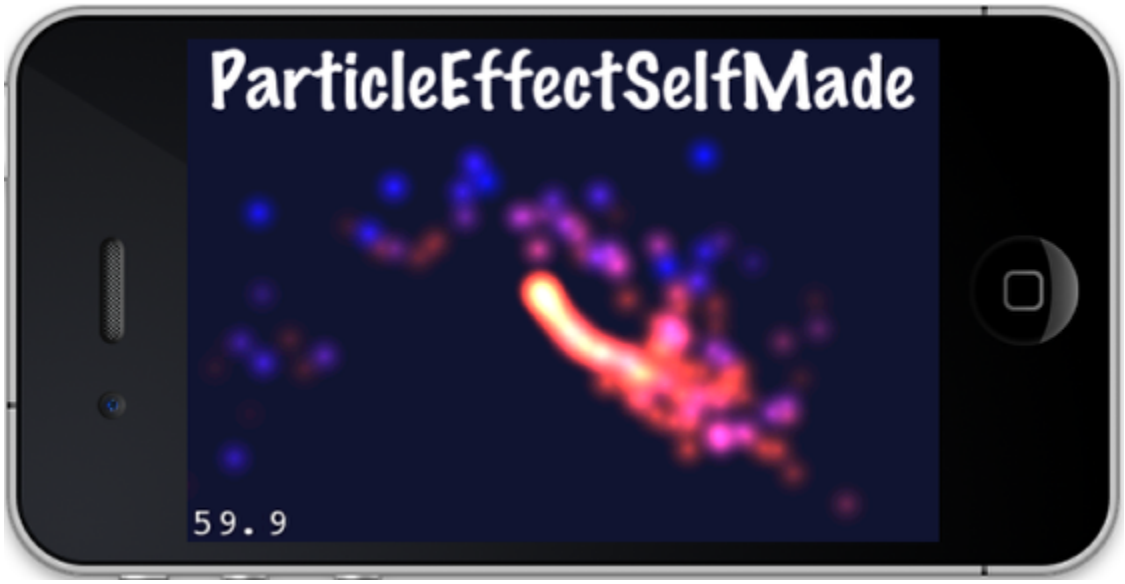
_IncomingNetworkConnections

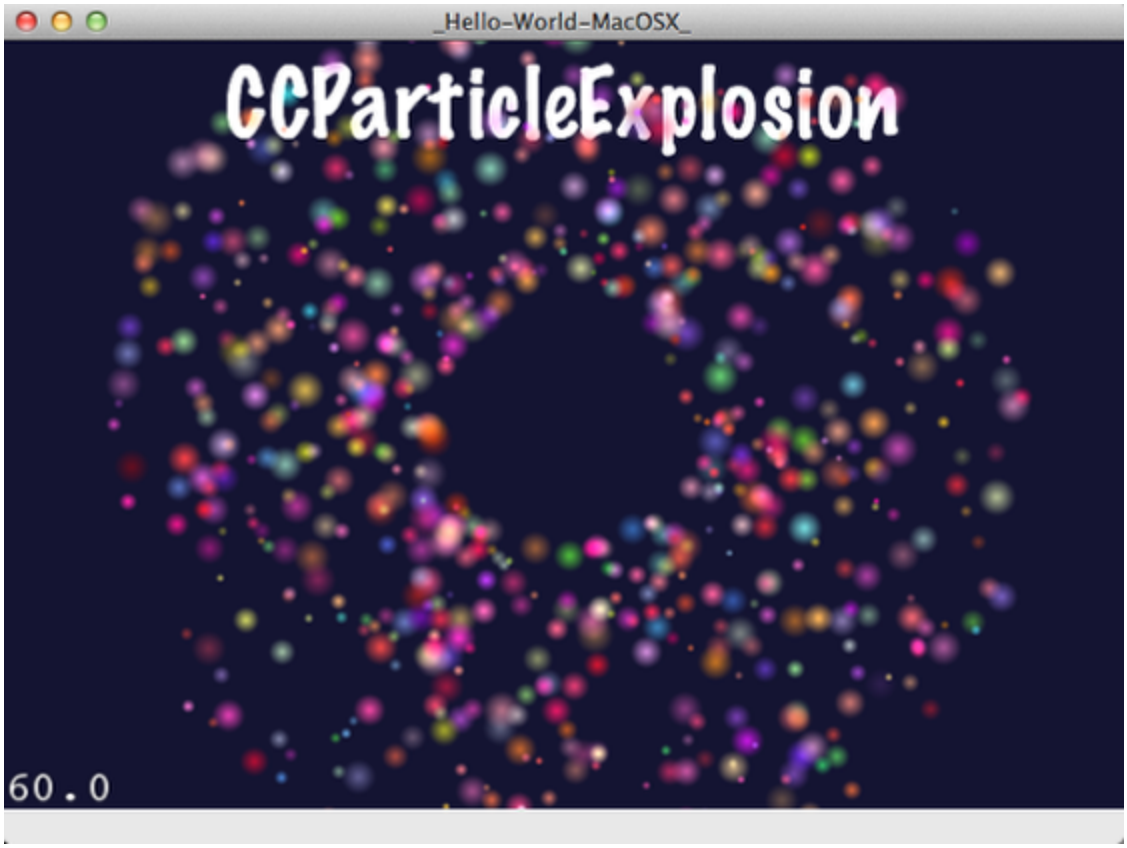
_Isometric-Tilemap-Template_Images

_Orthogonal-Tilemap-Template_Images

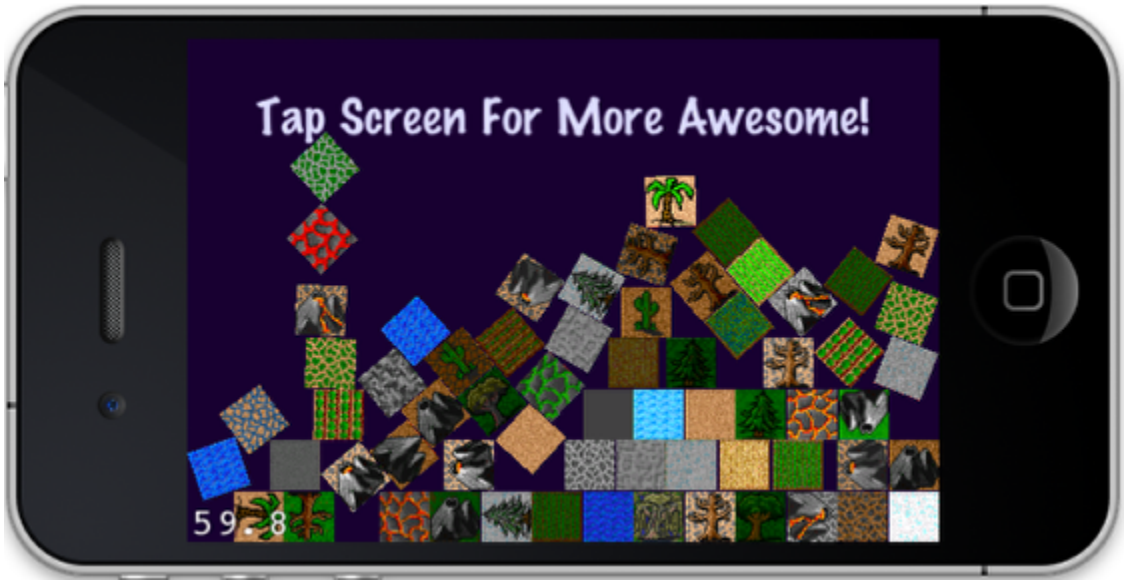
_Parallax-Side-Scroller-Template_Images

_Particle-Effects-Template_Images

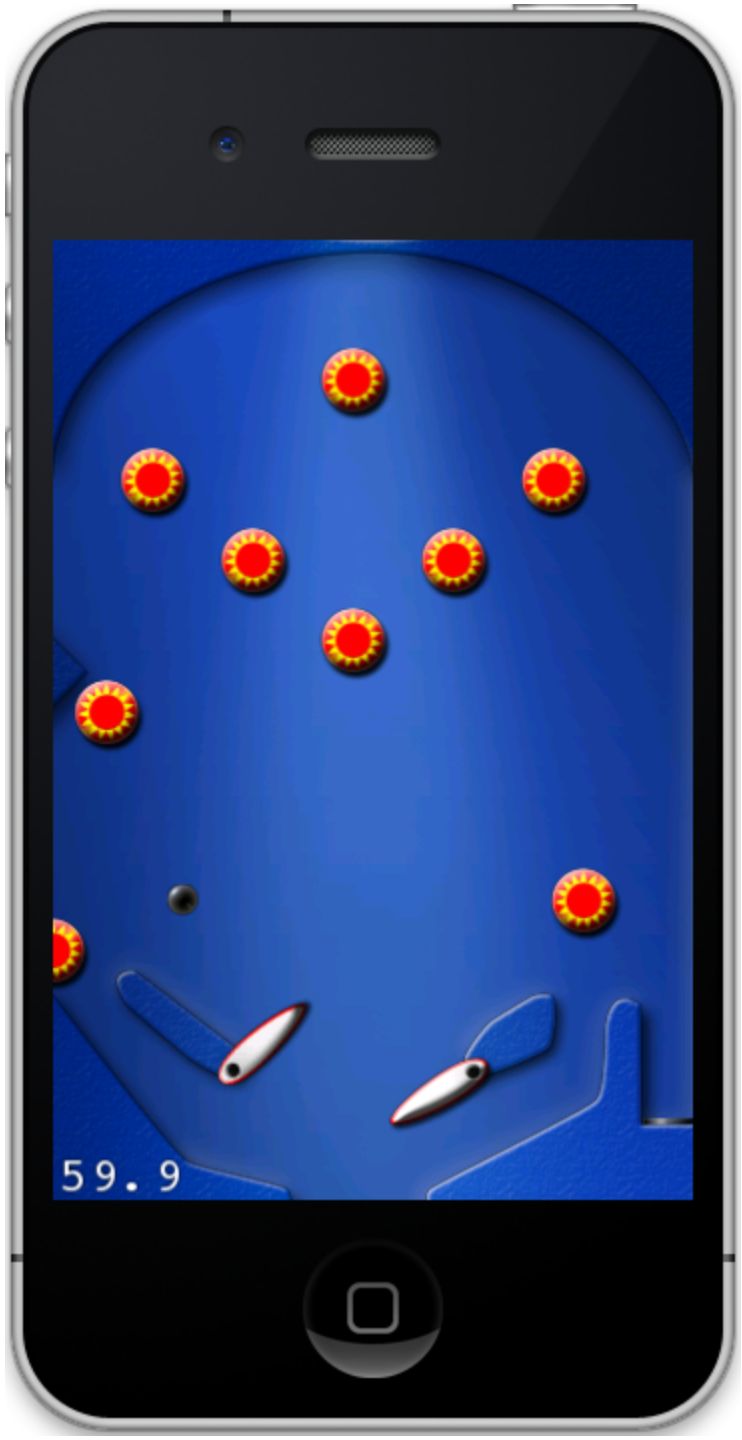


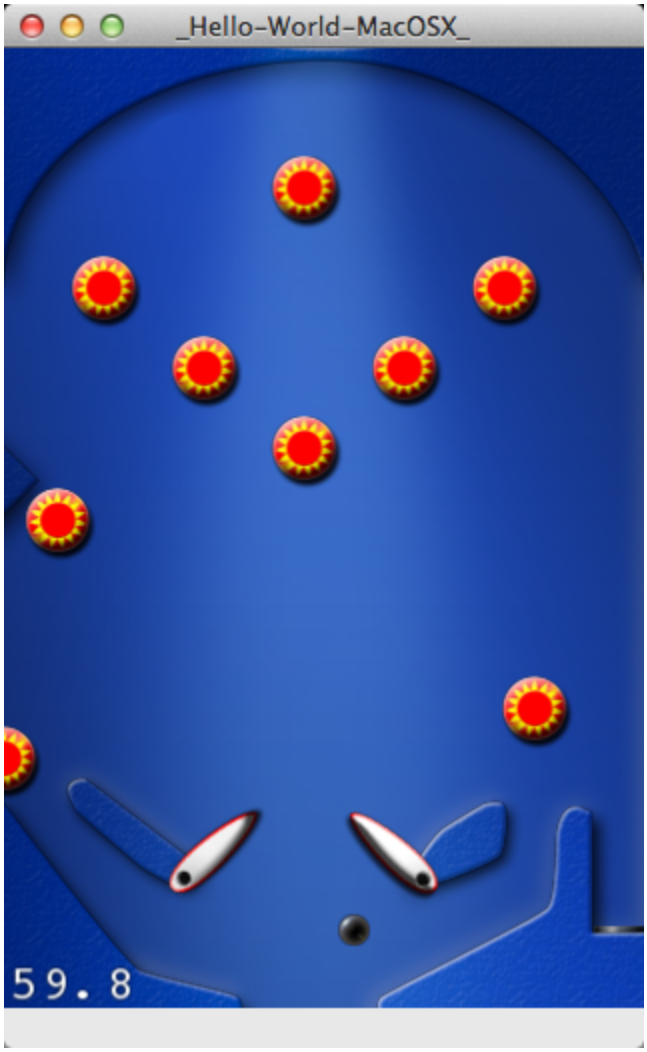


_Physics-Template_Images

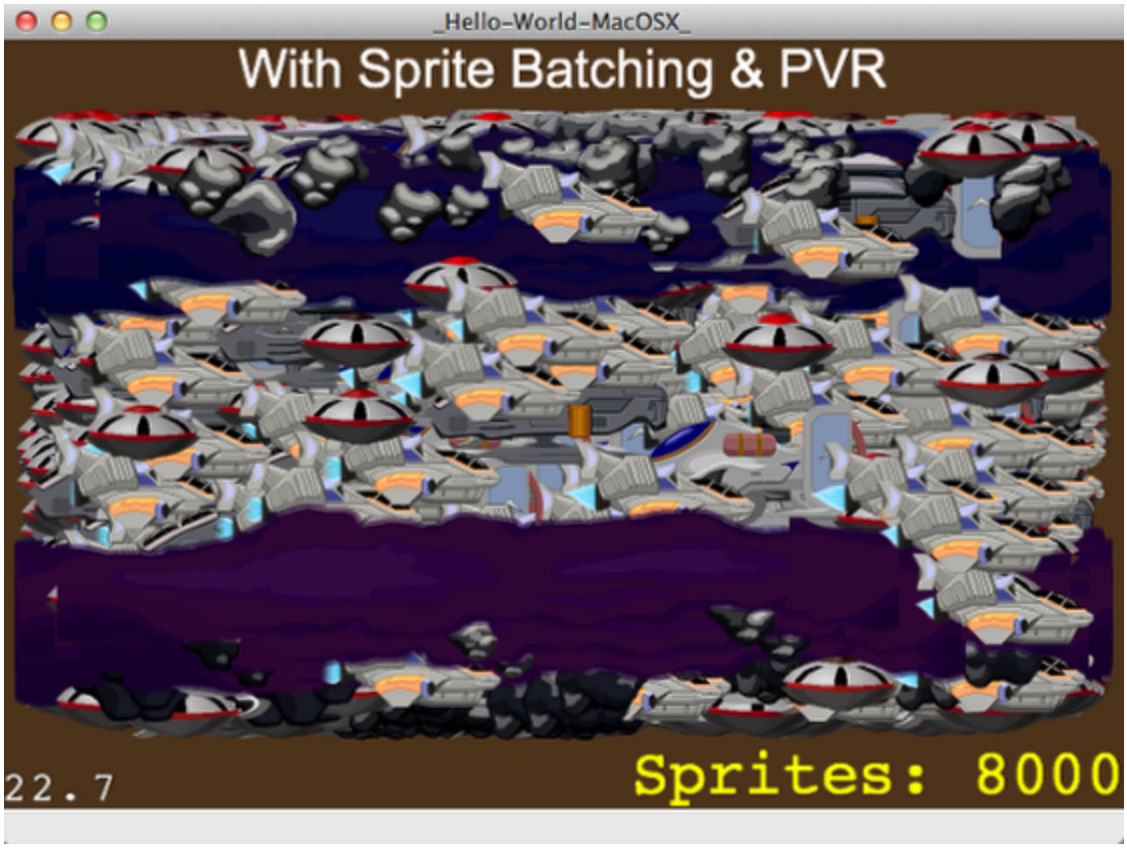


_Pinball-Game-Template_Images

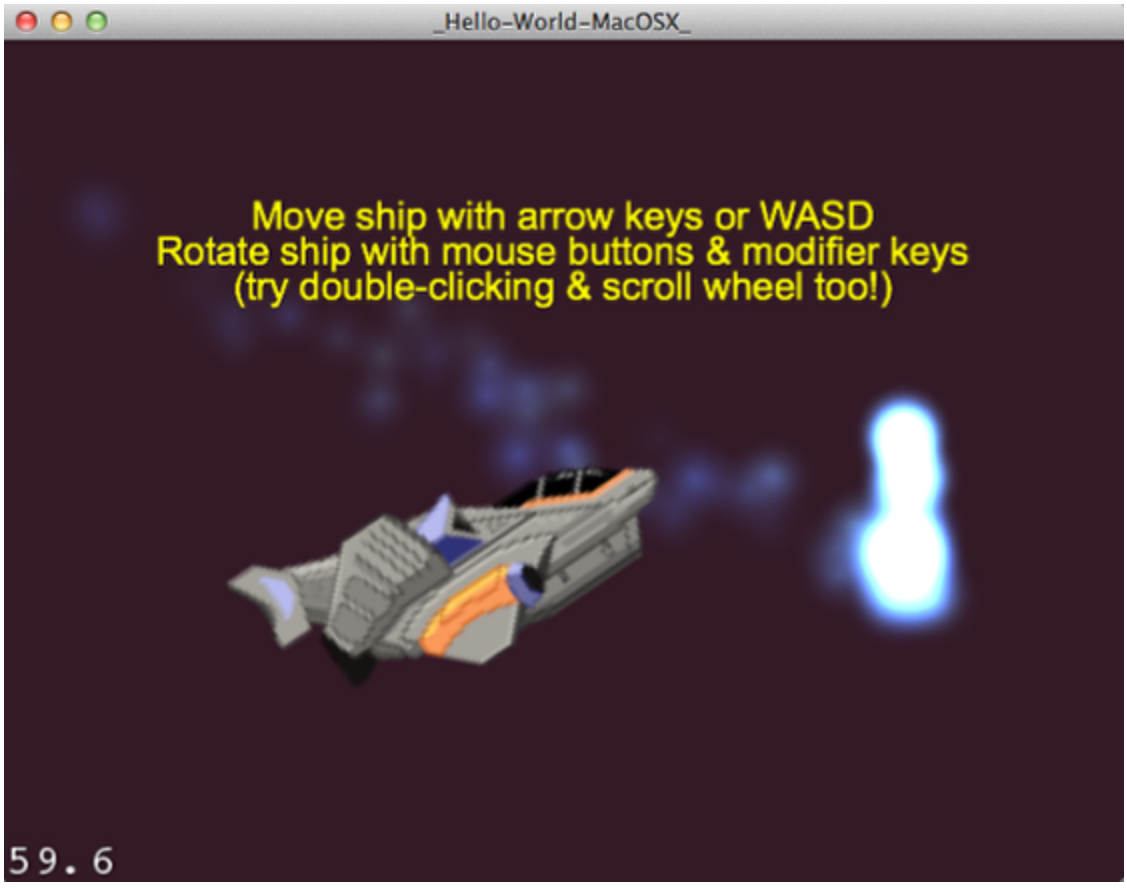

59.9

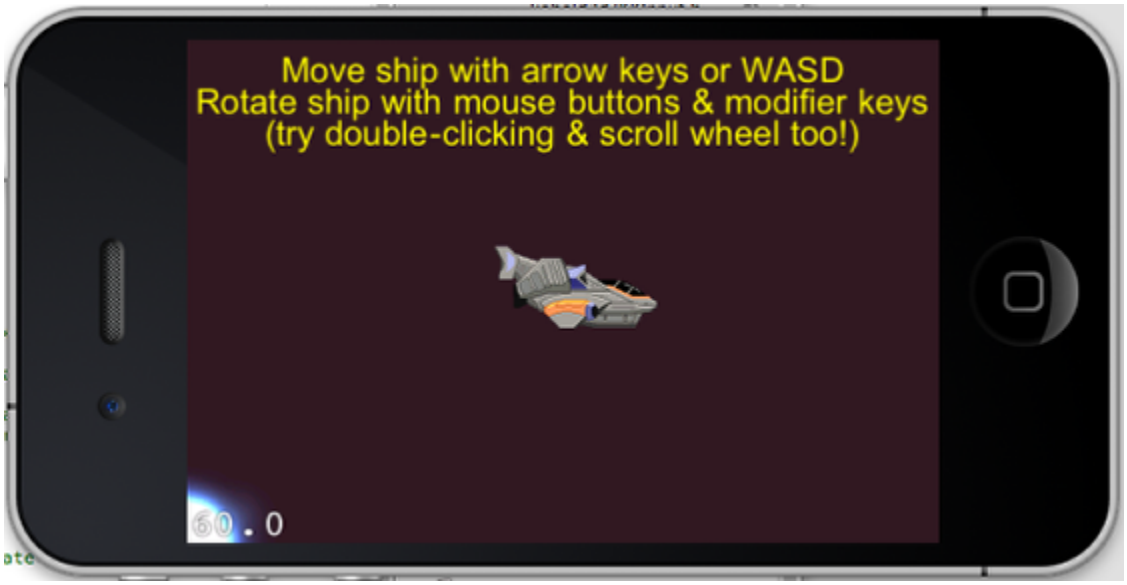


_Sprite-Performance-Template_Images



_User-Input-Template_Images





_TemplateProjects

- _AdvancedTemplateProjects
- _BasicTemplateProjects
- _GameTemplateProjects

_AdvancedTemplateProjects

Physics Box2D / Physics Chipmunk / Physics Chipmunk SpaceManager

iOS

Mac OS

The three physics template projects in Kobold2D all implement the same physics demo: adding dynamic bodies (boxes), collision callbacks, and the screen/window acting as an impenetrable barrier. A static body is used to attach three dynamic bodies which are connected via Revolute (Box2D) / Pivot (Chipmunk) joints to create a swinging "rope".

Chapter 12

If you're unsure which physics engine is right for you, check out these templates and use the one which you find easiest to work with. You might also want to read the FAQ entry: Which Physics Engine is the best?

Orthogonal Tilemap

iOS

Mac OS

The orthogonal tilemap project shows how to load a TMX tilemap created with the Tiled Map Editor. It demonstrates conversion of touches to tile coordinates, inspecting tile properties, the use of the Tiled Object Layer for trigger areas and scrolling of the tilemap.

Chapter 10

Isometric Tilemap

iOS

Mac OS

The isometric tilemap project shows how to load an isometric TMX tilemap created with the Tiled Map Editor. The isometric tilemap is drawn with depth buffering and 2D projection for correct Z ordering of tiles. The project demonstrates conversion of touches to tile coordinates, inspecting tile properties, the use of the Tiled Object Layer for trigger areas and scrolling of the tilemap. It includes a controllable player character. On Mac OS it illustrates how to process keyboard events.

Chapter 11

Sprite Performance

iOS

Mac OS

This project illustrates how to improve rendering performance by replacing sprites created from individual files to sprites created from sprite frame provided by a PVR compressed texture atlas. It also shows how to use CCSpriteBatchNode to improve performance even further.

This project can also be used for benchmarking sprite rendering performance of various sizes and image formats.

Blog Post

_BasicTemplateProjects

Hello Kobold2D

iOS

Mac OS

The standard template project for Kobold2D is the successor of the "Hello Cocos2D" template. It showcases Retina support, config.lua settings, OS and device detection, iAd banners, actions, built-in and custom font labels and sound.

Chapter 16

Use this template as a starting point if you want to start a Kobold2D project from scratch.

Hello Cocos3D

iOS

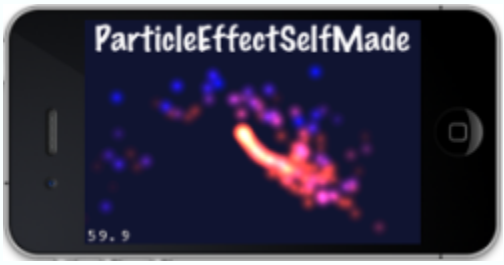
iOS

This template is similar to *Hello Kobold2D* and showcases the use of Cocos3D, mixed with regular Cocos2D nodes. Notice that Cocos2D nodes can be drawn in front of or behind 3D objects. Cocos3D currently does not support Mac builds.

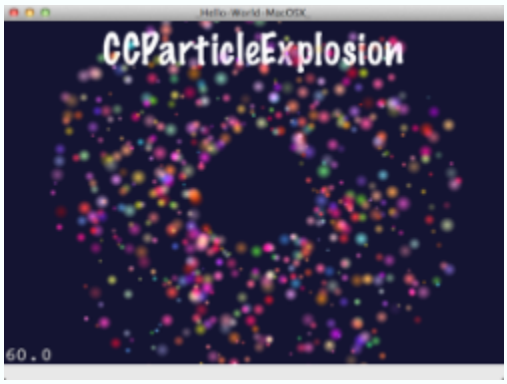
Chapter 16

Use this template as a starting point if you plan to work with Cocos3D. It has the required line `-force_load libcocos3d.a` in `BuildSettings-iOS.xcconfig` already set.

Particle Effects



iOS



Mac OS

This project shows you how to create particle effects by subclassing the CCParticleSystem class. It also displayed particle effects created with Particle Designer as well as Cocos2D's built-in particle effects.

Chapter 9

User Input

iOS

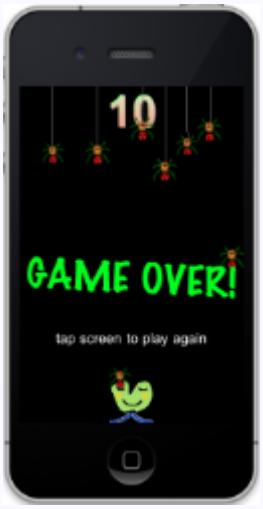
Mac OS

Illustrates how to use the KKInput class which wraps user input into a collection of convenience method. The main difference being that KKInput allows you to poll the input state at any time from any class without having to register to receive input events. KKInput takes the grunt work out of input processing, giving you the ability to easily test for double-clicks, scroll wheel & mouse position changes, key states with or without modifiers, and more.

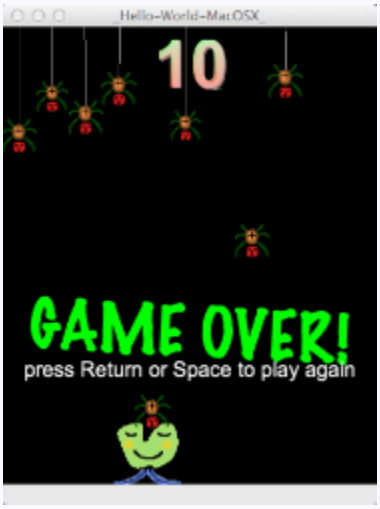
Exclusive Kobold2D Feature!

_GameTemplateProjects

Doodle Drop



iOS



Mac OS

Doodle Drop is a fully playable game in which you attempt to avoid the dropping spiders by tilting your device. The game showcases accelerometer input (keyboard input on Mac), sound, music, a game over screen, a custom loading screen, score counting wth a bitmap font label, object pooling to avoid frequent alloc/dealloc cycles and advanced use of easing, rendering lines, simple collision testing, sequence and call func actions.

Chapter 4

Parallax Side-Scroller

iOS

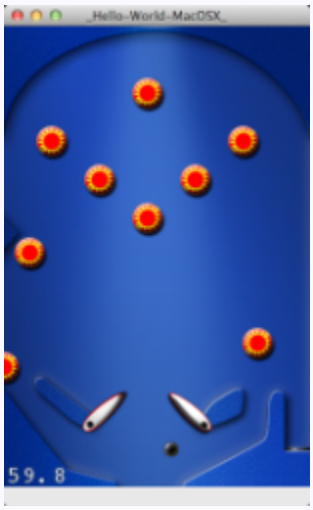
(no Mac OS version)

The Parallax Side-Scroller is a classic shoot'em up game with a virtual joypad and button provided by SneakyInput. The project shows you how to create an endlessly scrolling parallax background, how to structure bigger projects into multiple classes, how to pool and re-use game objects, how to keep nodes within a defined area by overriding setPosition, how to create reusable game components, how to turn your game's scene into a Singleton to make it accessible to other classes, and how to access other classes through your game's scene.

Chapters 6-8

Pinball Game

iOS



Mac OS

The Pinball project is a functional pinball table with bumpers, flippers, a plunger and a ball. Box2D is used for physics simulation, the collision shapes were created with PhysicsEditor. The plunger and flippers are animated via Box2D motors. Collision events are forwarded via Objective-C messages to individual objects and correspondingly named selectors (eg beginContactWithBall, endContactWithPlunger). Includes a demonstration of gravitational pull by uncommenting the function applyForceTowardsFinger.

Chapter 13

Kobold2D

The Kobold2D Kobold2D

The Kobold2D Kobold2DKobold2DKobold2D Release Notes

The Kobold2D FAQ

Kobold2D

Kobold2D

- Xcode 4.0
- iOS 4.3
- Mac OS X Snow Leopard (10.6)
- 500 MB

- Xcode 4.1
- Mac OS X Lion (10.7)
- Objective-C

Kobold2D

Kobold2D /
!! Kobold2D

1. Kobold2D
2. **Kobold2D.pkg**

3.
4. Kobold2D!!

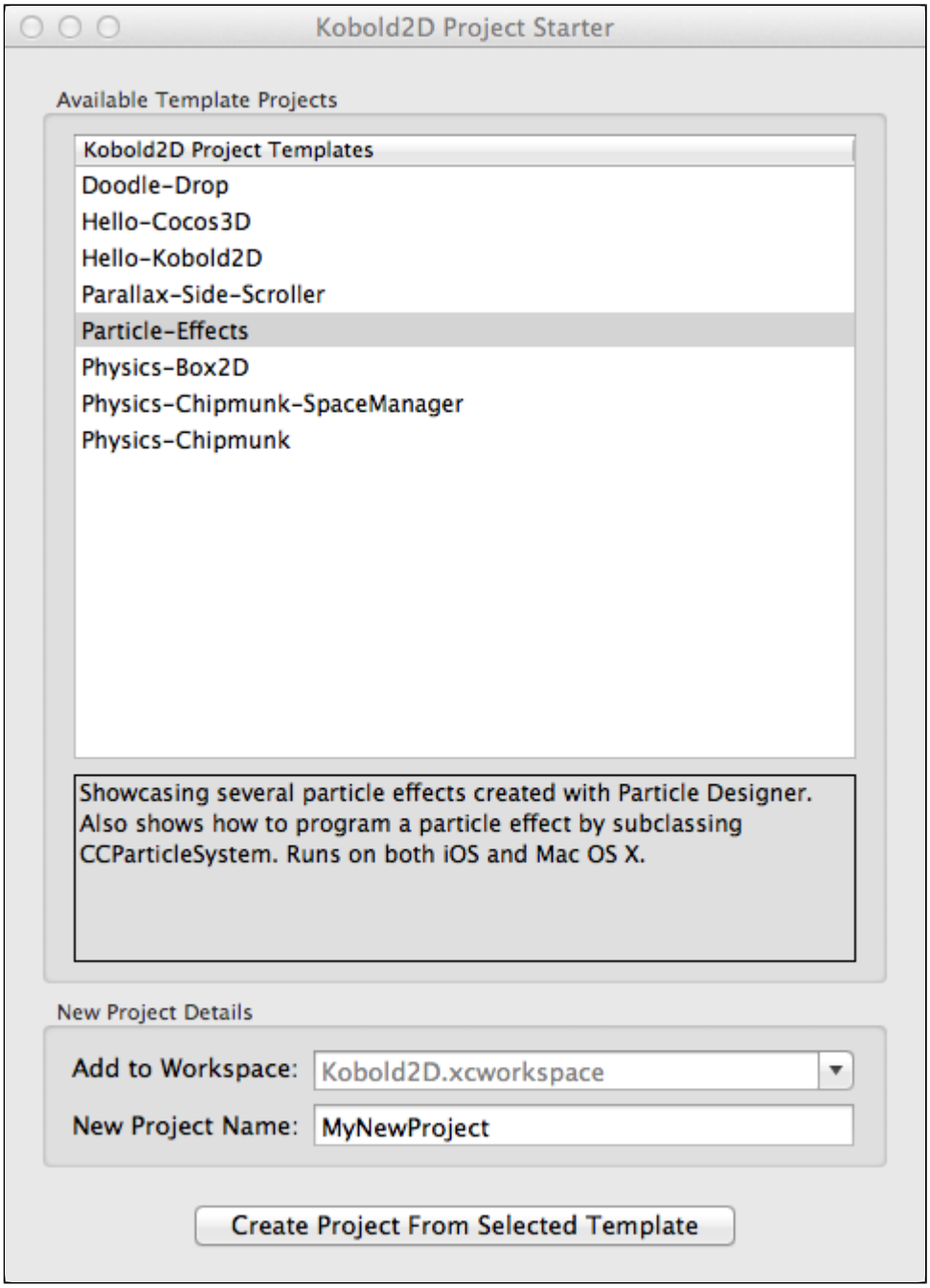
Kobold2D:

```
~/Kobold2D/
```

~/Kobold2D/Applications/Kobold2D

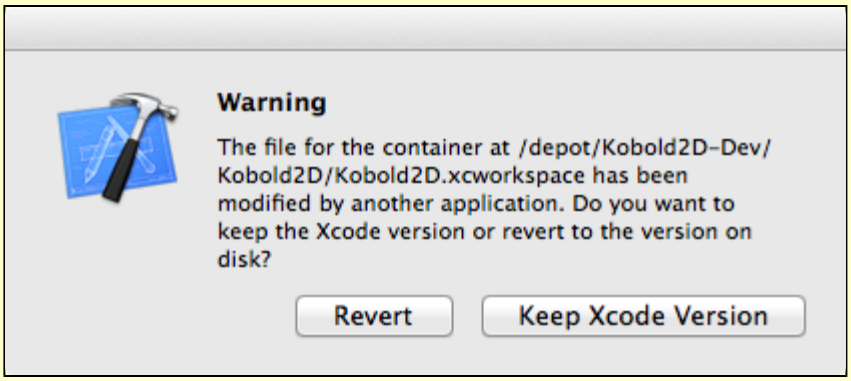
Kobold2D

Kobold2D*Kobold2D Project Starter.app*~`Kobold2D\Kobold2D-X.Y` (X.YKobold2D)



1. *Kobold2D Project Starter.app*:
2. 1
3. :
    a. *Add to Workspace*
    b. *Add to Workspace*
4.
5. *Create Project From Selected Template*
6. Xcode
7. .

Xcode
Xcode



Xcode:
*Keep Xcode Version*Kobold2DXcodeProject Starter tool*Revert*

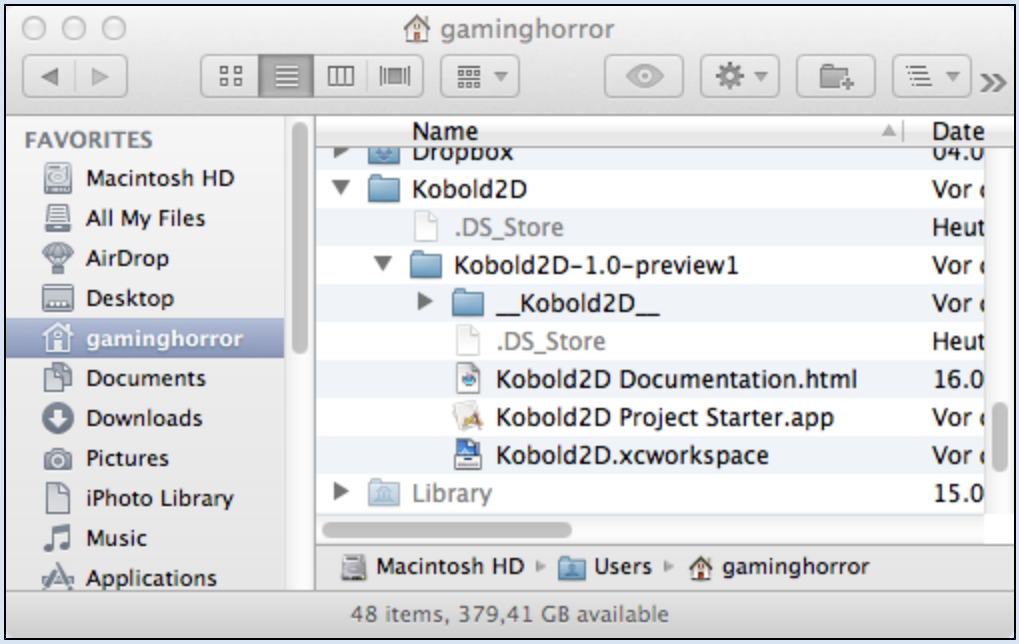
Xcode
XcodeKobold2D(*File -> New -> New Project...*

Kobold2D

Kobold2D

Kobold2D

Kobold2DKobold2D:



`__Kobold2D__` Kobold2DKobold2D `__Kobold2D__`

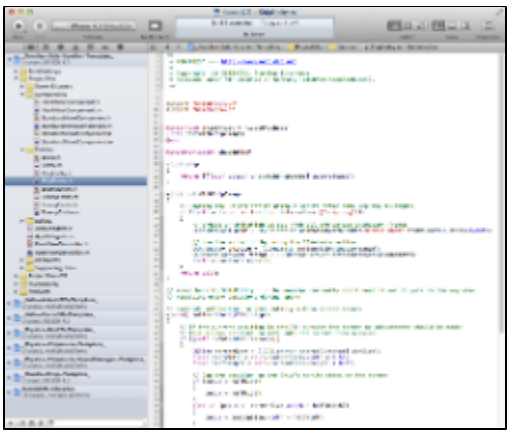
Kobold2D

`~/Kobold2D/Kobold2D-X.Y` `Kobold2D.xcworkspace` Xcode 4XcodeKobold2D
You can also create your own workspace.

Kobold2D

Kobold2DKobold2DKobold2DKobold2D

Kobold2DKobold2D

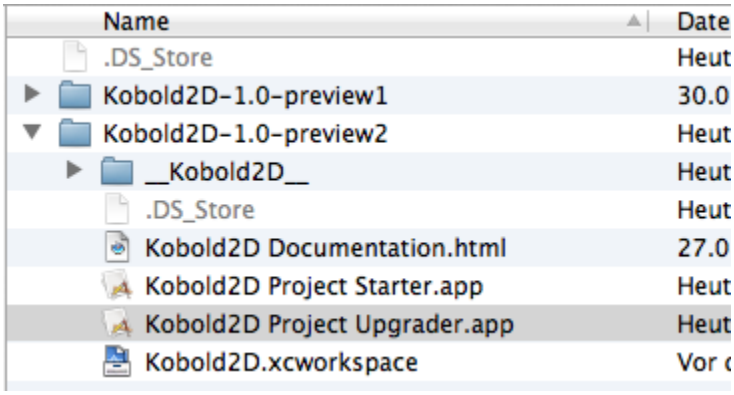


Kobold2D21

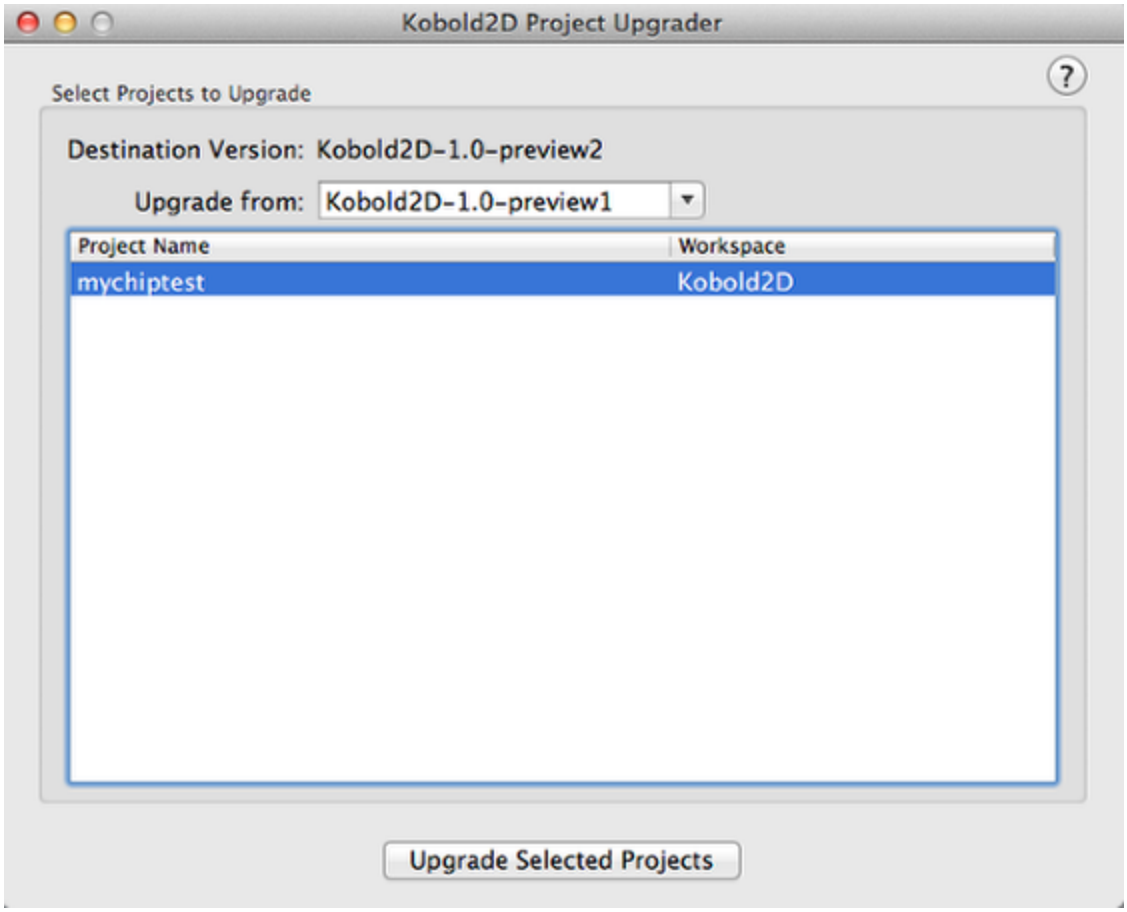
Upgrading Kobold2D Projects

> Kobold2D

Run the Kobold2D Project Upgrader.app



Kobold2D
Kobold2D Project Upgrader.app



Kobold2D Project Upgrader.app Kobold2D

FinderCmd+Shift+

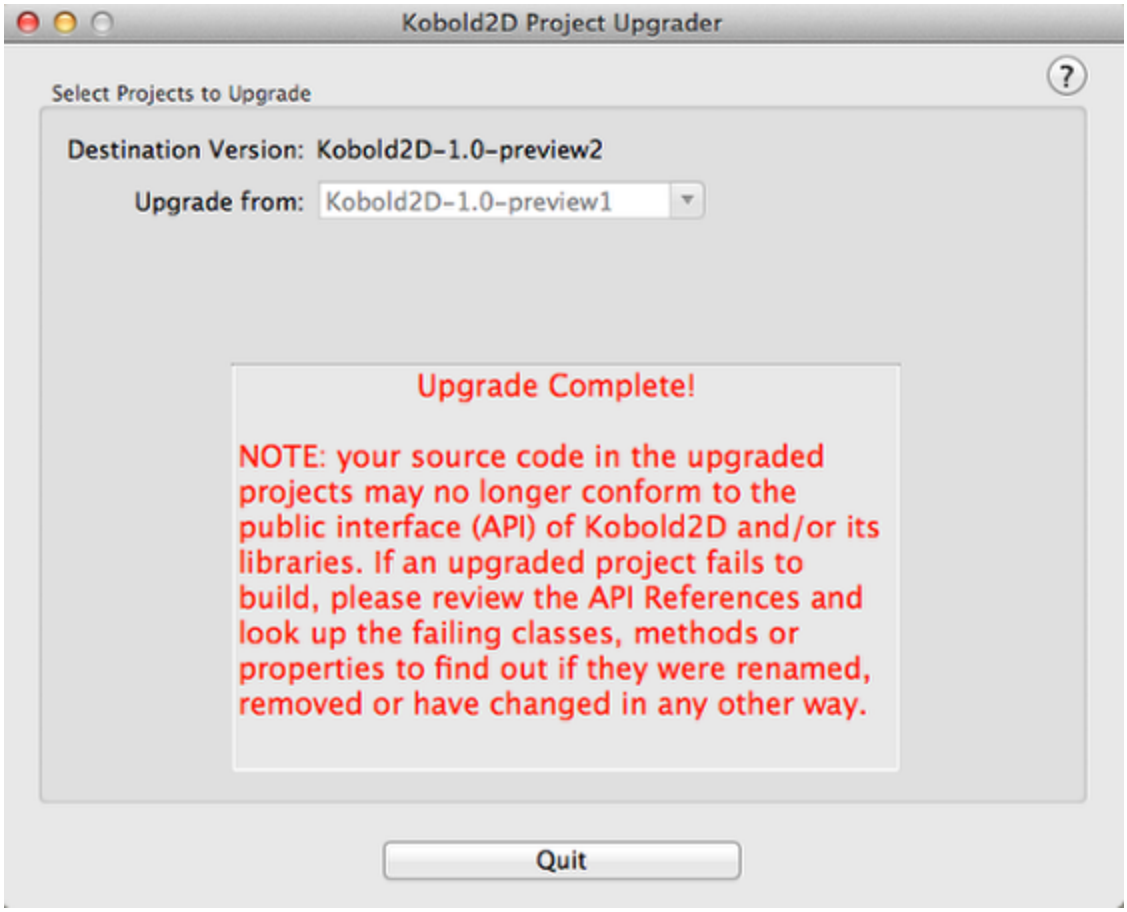
Upgrade from Kobold2DKobold2D

If a project is not in the list...
The **Kobold2D Project Upgrader.app** tries to ensure that you don't accidentally overwrite already existing projects. For that reason not all projects may be listed. In particular projects for which a folder with the same name already exists in the destination version's folder will not be shown to prevent accidentally overwriting that project.

In a few cases this may not be an already upgraded project. Assuming you have already created a project with the same name (eg `test`) in the preview 2 version and then want to update the older, different but also named `test` project from preview 1, you will first have to rename or delete either one of these two `test` projects before you can upgrade the old `test` project.

Click "Upgrade Selected Projects"

Close Xcode!
It is recommended to close Xcode before clicking the **Upgrade Selected Projects** button. At the very least you should close the workspaces and projects that you are going to upgrade.



The upgrader tool will copy the project folder and its contents over to the destination version, in this case `Kobold2D-1.0-preview2`. It will also upgrade the project's workspaces.

The informational message is a reminder that changes to the API may cause build errors in the upgraded projects, which are not related to the upgrade process itself. The **Kobold2D Project Upgrade.app** can not modify your source code to make it compatible with the latest APIs. That is always a manual process, but in most cases the necessary changes are minimal. Please refer to the Kobold2D Release Notes for information on API changes.

You can now open the workspaces in the destination Kobold2D version and continue working.

> **Be aware of which workspace version Xcode opens!**
> Keep in mind that Xcode automatically re-opens the most recently used workspaces and projects. If you do not manually open the corresponding workspace(s) from the **latest Kobold2D version** you will **continue to work with the second to last version of Kobold2D**, which is probably not what you want.

Migrating a Cocos2D Project

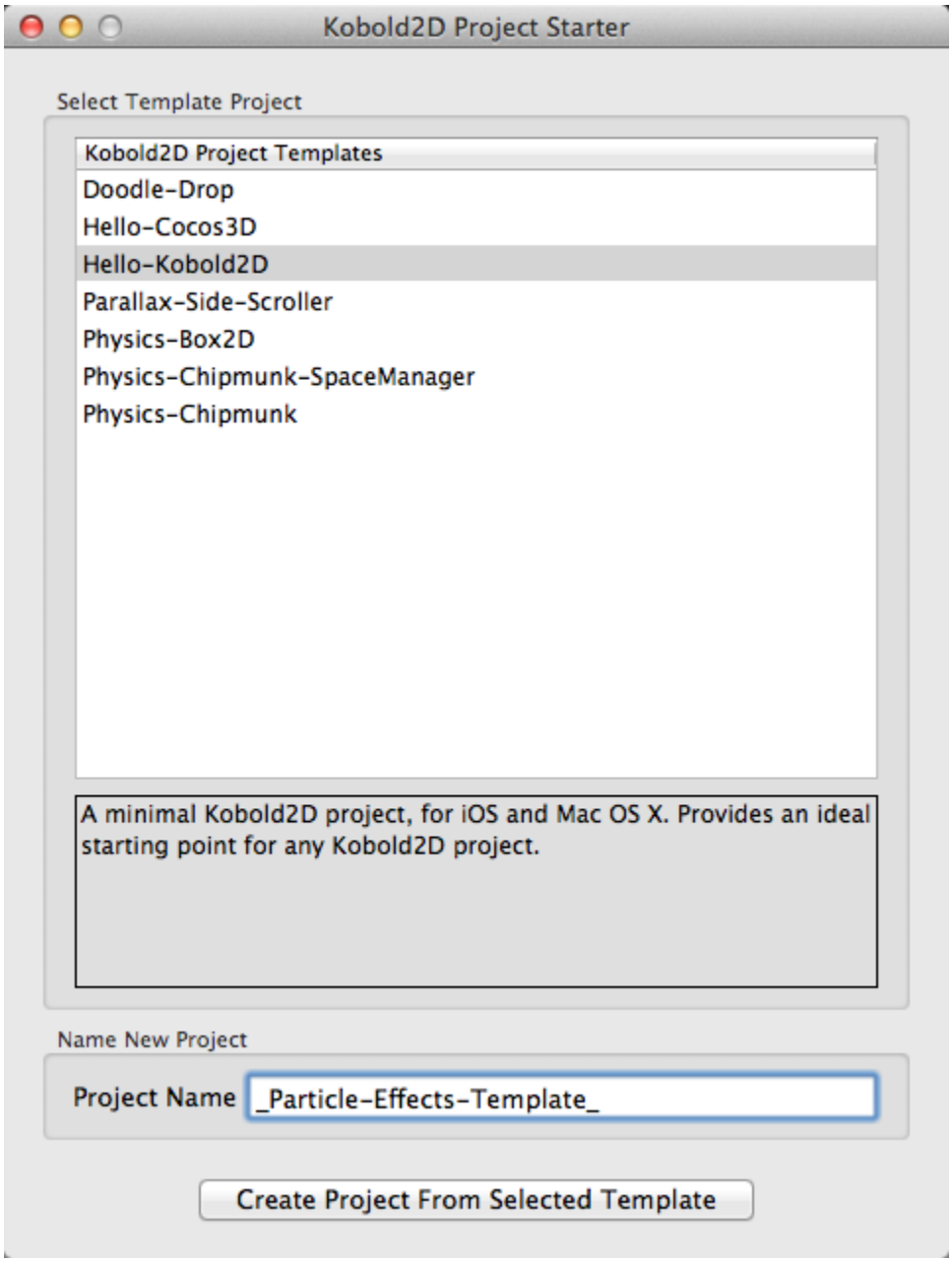
This guide explains by example how an existing Cocos2D project can be converted to a Kobold2D project. In this step by step guide the project **ParticleEffects03** from the Learn Cocos2D book will be migrated to Kobold2D.

> This is a general guide to explain the process of converting an existing Cocos2D project to Kobold2D. This guide can not offer a solution to every possibly required change to successfully migrate every project. Each project is different and may require custom modifications or experience issues not outlined in this guide. Please use the Kobold2D Forum if you need additional help on migrating a project to Kobold2D.

If your project is nearing completion it may be better to finish that project as is. You won't benefit much from converting an almost complete project to Kobold2D and you risk running into subtle and time-consuming issues that might delay the project's release. Instead just start your next project straight as a Kobold2D project.

As you migrate from your Cocos2D project to a Kobold2D project, be aware that the third party libraries provided by Kobold2D (including but not limited to: cocos2d-iphone, cocos3d, Box2D, Chipmunk, etc) may be of a different version. Be prepared to adapt your code accordingly.

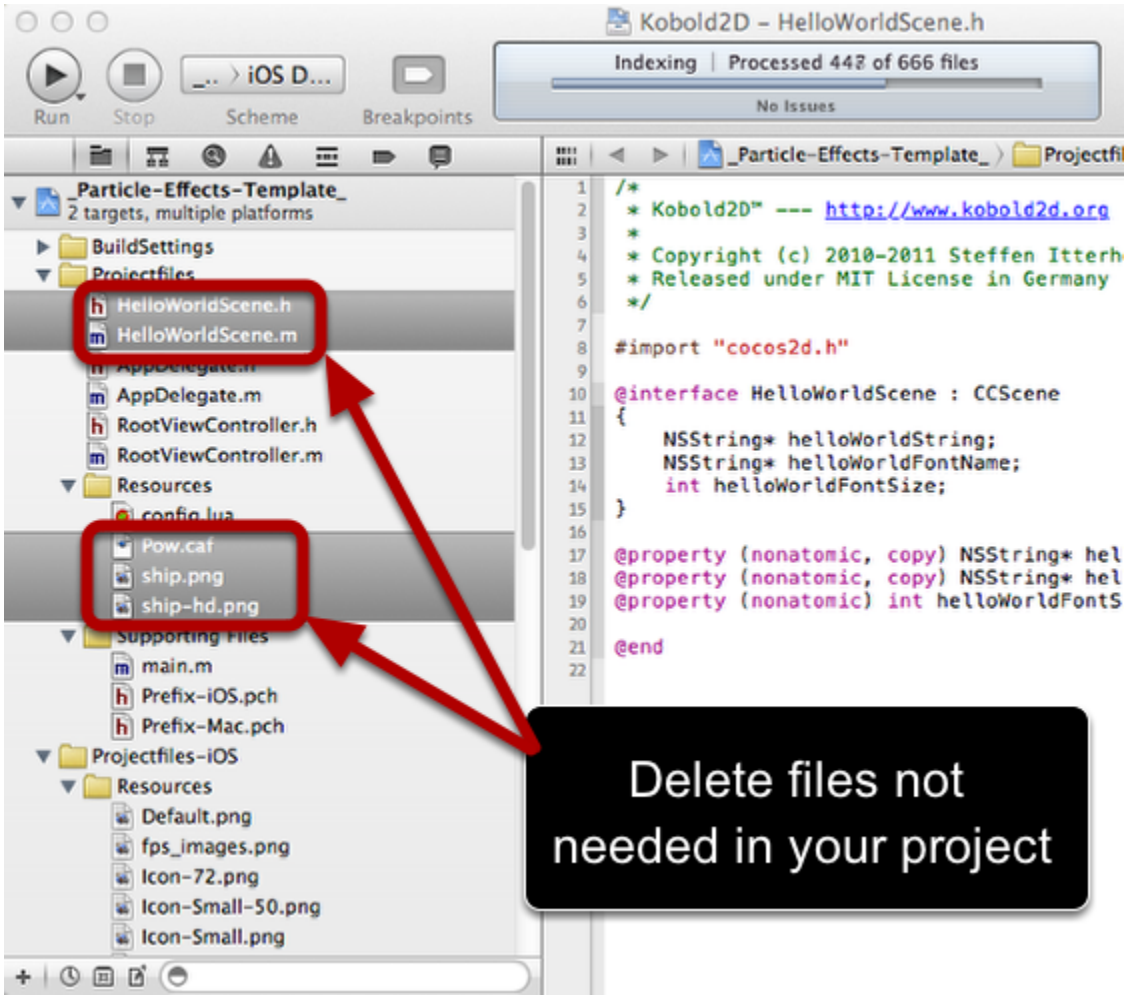
Create a Kobold2D Project



The first step is to create a new Kobold2D project from one of the templates. It is recommended to use the Hello Kobold2D template even if your project uses a physics engine. The Hello Kobold2D project is already setup for both physics engines.

The new project is named _**Particle-Effects-Template**_ because the goal is to turn it into a Kobold2D template project. You should use an appropriate name for your new project.

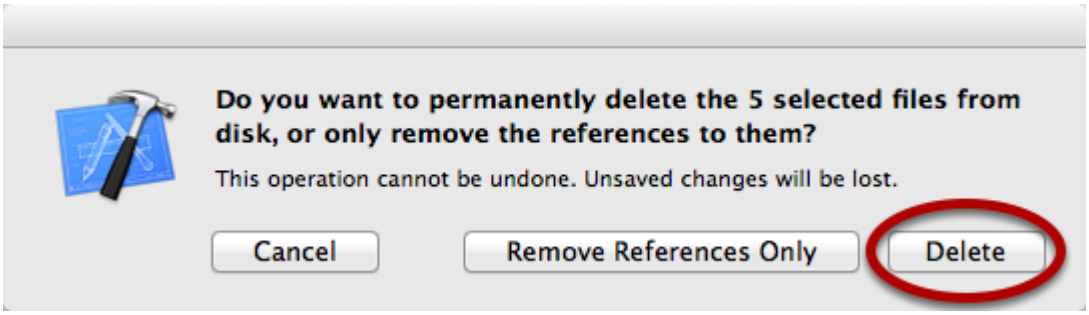
Cleanup the new Kobold2D Project



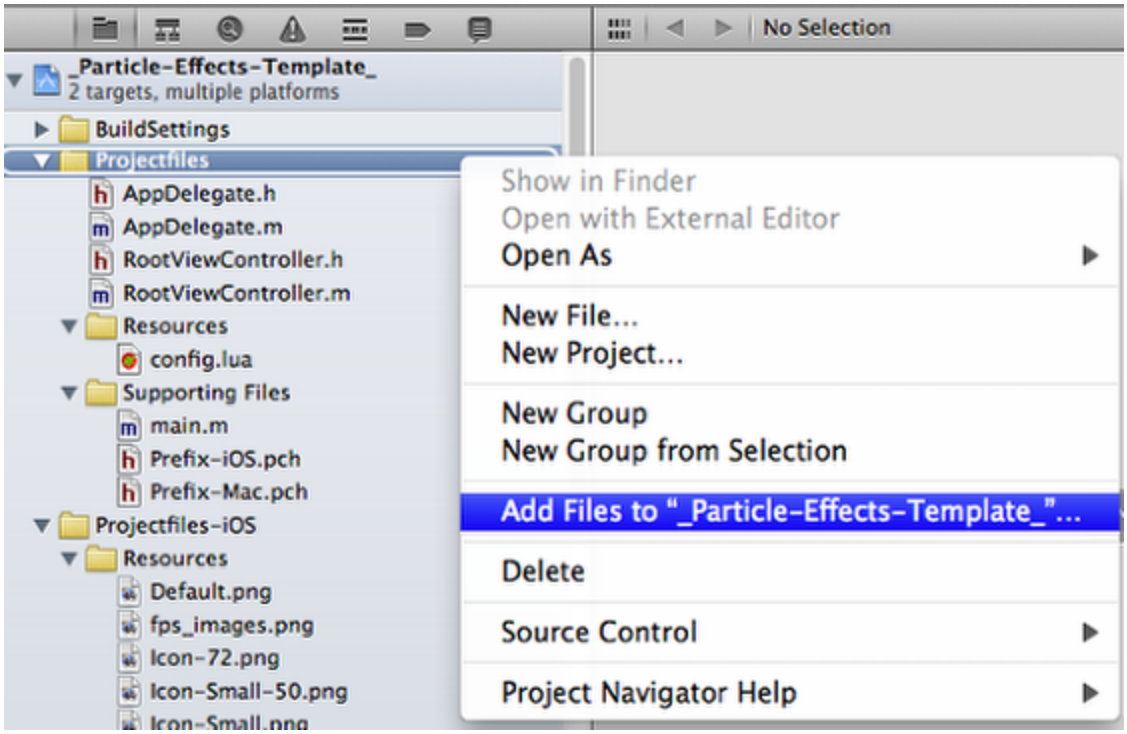
There are a few files which won't be needed by your project. This includes the HelloWorldScene class and several resource files (Pow.caf, ship.png, ship-hd.png). Under Projectfiles-iOS delete the kobold* font files. Future versions of the Hello Kobold2D template project may add additional resources that you may want to remove.

Keep config.lua
Do not delete the file **config.lua**! Kobold2D projects require it.

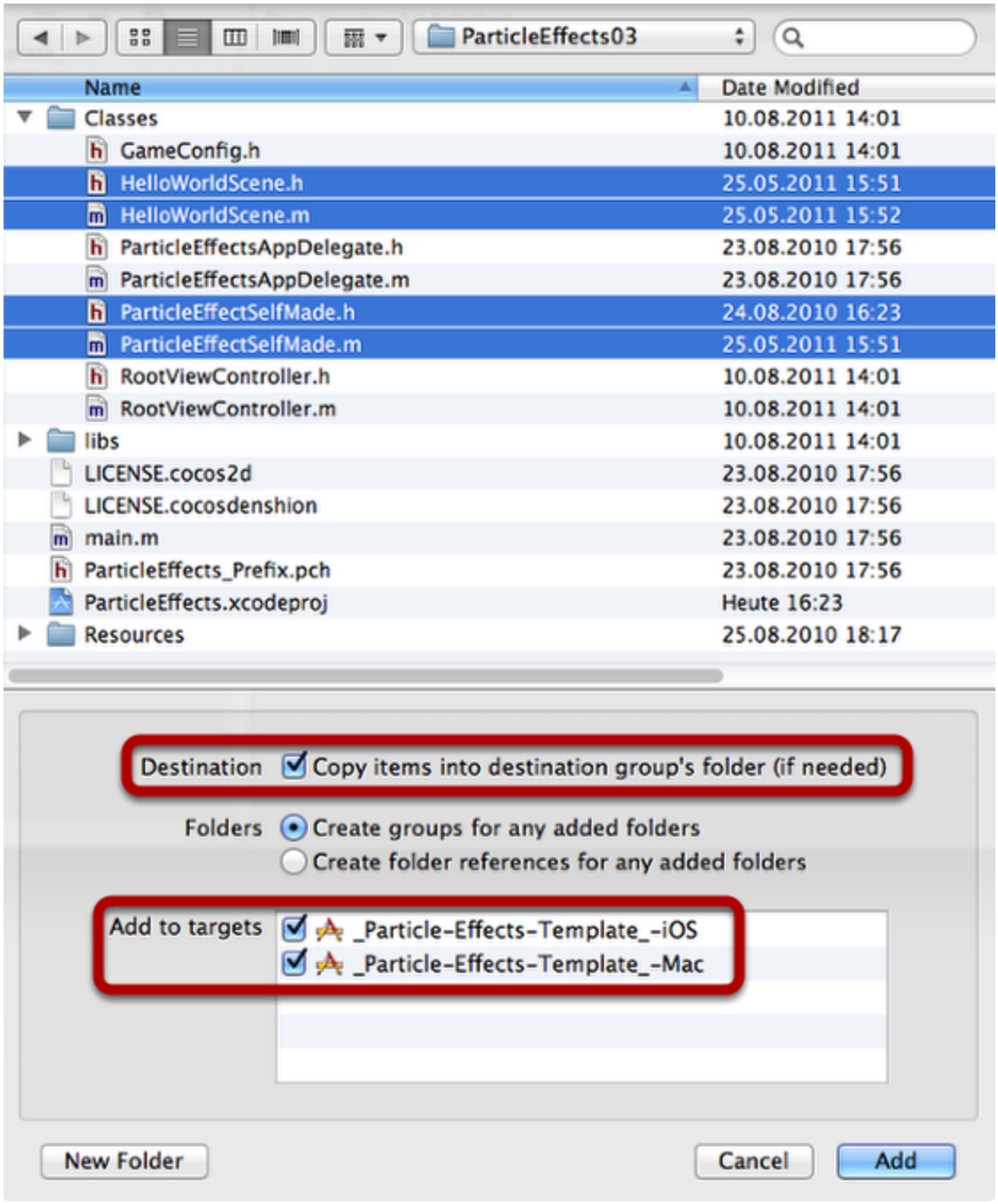
You will want to **Delete** these files to also remove them from the file system:



Add your project's source code



Select the **Projectfiles** group, right-click or option-click to bring up the menu. Then select **Add Files to "projectname"...**. This will bring up the following Add Files dialog:



Locate your original project's source code files. Be sure to select all of your source code, but only the source code that you wrote.

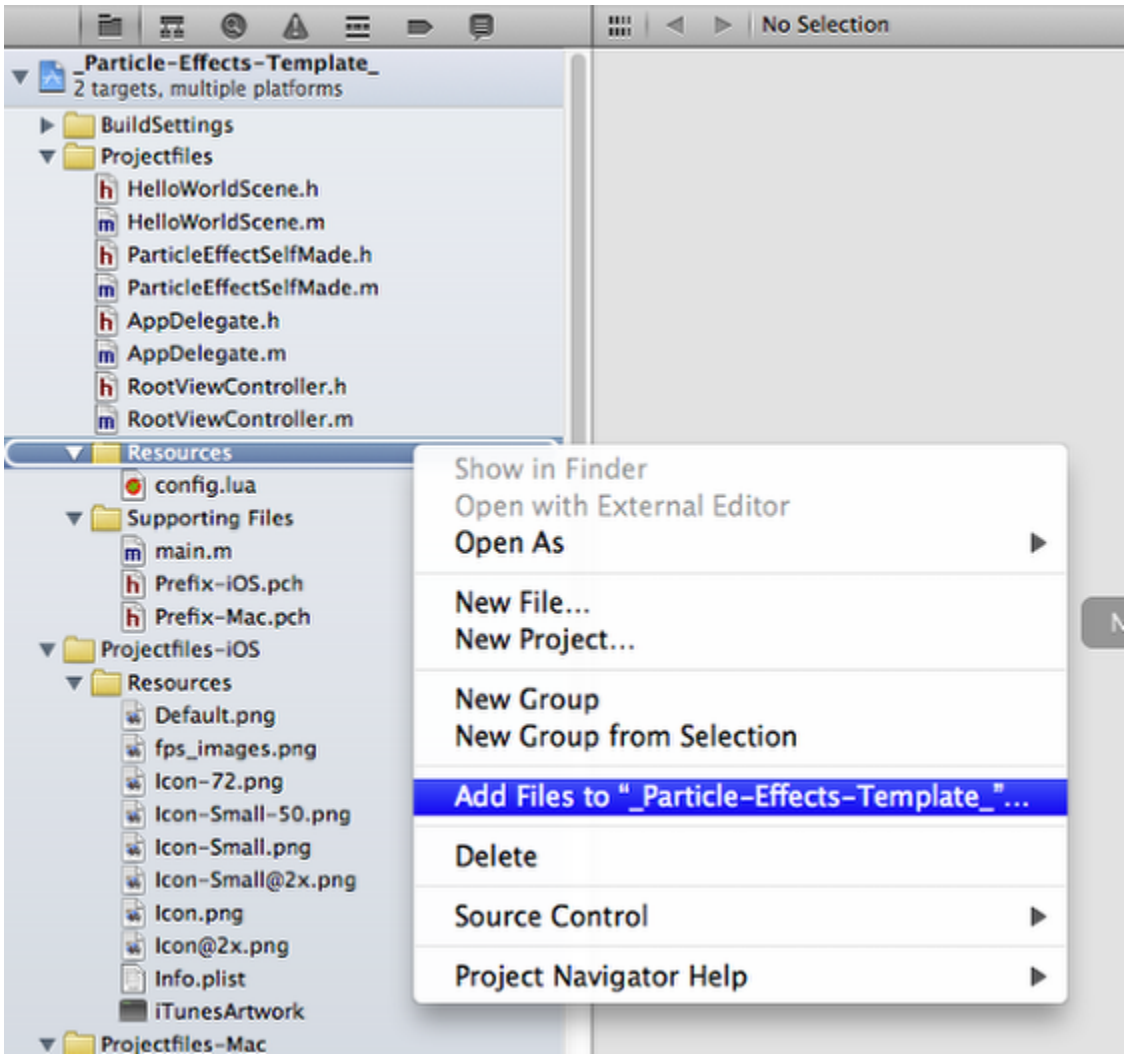
In particular, make sure you **do not include** the following files:

- GameConfig.h
- (ProjectName)AppDelegate.*
- RootViewController.*
- main.m
- (ProjectName)_Prefix.pch
- any library source code, such as: cocos2d, box2d, chipmunk, etc. source code files

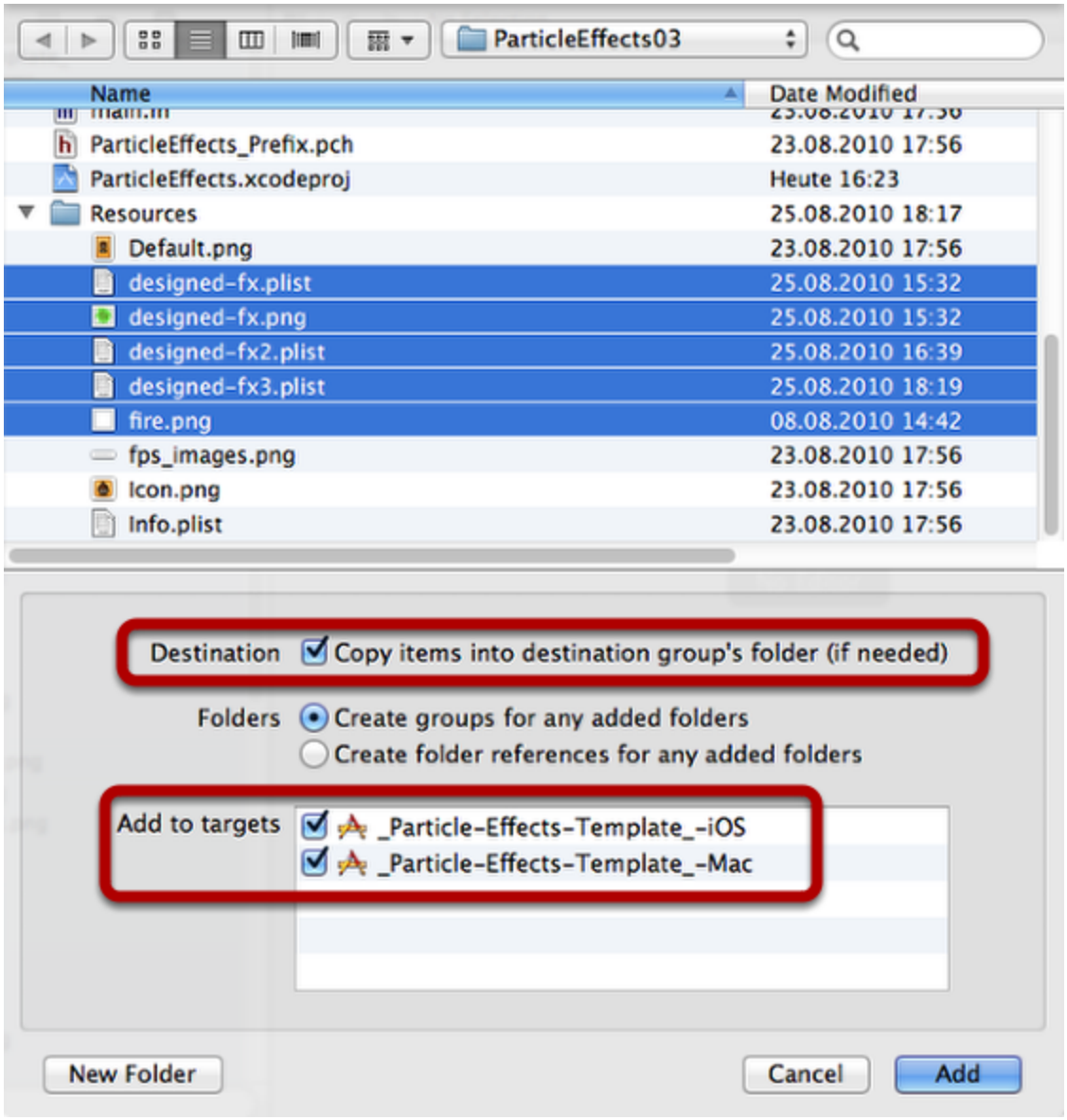
In case you accidentally added one of the above files you should select and delete them from the Kobold2D project. Just make sure you delete those you added and not the ones that may have already existed in the Kobold2D project.

Before clicking **Add** make sure you've selected the checkbox **Copy items into destination group's folder (if needed)** and the corresponding targets that you want to support (if in doubt select both).

Add your project's resource files



Now select the Resources group and option-click or right-click. Once again select **Add Files to "ProjectName"...** to bring up the Add Files dialog again, this time for adding your project's resource files:



Select the resource files that are unique to your project, eg the ones that you created and added.

Do not include the standard resource files! In particular, make sure you **do not include** the following files:

- Info.plist
- Default.png, Icon.png or any variant (Default@2x.png, Icon@2x.png, etc.)

In case you accidentally added one of the above files you should select and delete them from the Kobold2D project. Just make sure you delete those you added and not the ones that may have already existed in the Kobold2D project.

If you have modified any of the default resource files above, make a note (preferably not a mental note as you'll likely forget - you are human, right?). That note should read:
Remember to copy my project's modified standard resource files over the ones provided in the Kobold2D project, using Finder.
Also remember to check the Info.plist file for any modifications/additions that I need to copy from my original project to the new Kobold2D project's Info.plist file.

About Info.plist
You should be aware that the Hello Kobold2D project contains two Info.plist files, one for iOS apps and one for Mac OS X apps. If you plan to support both platforms you may need to modify both. If you don't, make sure you modify the correct one. Platform-specific resource files are located in the **Projectfiles-iOS** respectively **Projectfiles-Mac** folders.

Before clicking **Add** make sure the Add Files settings are correct. The checkbox **Copy items into destination group's folder (if needed)** must be checked as well as the corresponding targets that you want to support (if in doubt select both).

Modify the config.lua


```
local config =
{
    KKStartupConfig =
    {
        -- load first scene from a class with this name
        FirstSceneClassName = "HelloWorldScene",

        -- set the director type, and the fallback in case the first isn't available
        DirectorType = DirectorType.DisplayLink,
        DirectorTypeFallback = DirectorType.NSTimer,

        MaxFrameRate = 60,
        DisplayFPS = YES,

        EnableUserInteraction = YES,
        EnableMultiTouch = NO,

        DefaultTexturePixelFormat = TexturePixelFormat.RGBA8888,
        GLViewColorFormat = GLViewColorFormat.RGB565,
        GLViewDepthFormat = GLViewDepthFormat.DepthNone,
        GLViewPreserveBackBuffer = NO,
        GLViewMultiSampling = NO,
        GLViewNumberOfSamples = 0,

        Enable2DProjection = NO,
        EnableRetinaDisplaySupport = YES,

        DeviceOrientation = DeviceOrientation.Portrait,
        AutorotationType = Autorotation.UIViewController,
        ShouldAutorotateToLandscapeOrientations = YES,
        ShouldAutorotateToPortraitOrientations = YES,
        AllowAutorotateOnFirstAndSecondGenerationDevices = NO,

        -- iAd setup
        -- Note: for iAd to support autorotation you should use: AutorotationType =
        ViewController
        EnableAdBanner = YES,
        LoadOnlyPortraitBanners = NO,
        LoadOnlyLandscapeBanners = NO,
        PlaceBannerOnBottom = NO,

        AutoScale = YES,
        AcceptsMouseMovedEvents = NO,
        WindowFrame = RectMake(1024-640, 768-480, 640, 480),
    },

    -- you can create your own config sections using the same mechanism and use KKConfig to
       access the parameters
    -- or use the KKConfig injectPropertiesFromKeyPath method
    HelloWorldSettings =
    {
        HelloWorldString = "Hello Kobold2D!",
        HelloWorldFontName = "Marker Felt",
        HelloWorldFontSize = 50,
    },
}
```

Now open the file **config.lua** in the new Kobold2D project. You will find it in the Projectfiles/Resources group. The above screenshot gives you hints as to what you may want to or need to change.

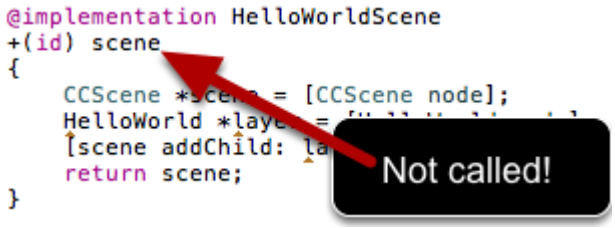
At the very least you will have to change the FirstSceneClassName parameter to the name of the CCScene or CCLayer class that Kobold2D should load as the very first scene. You may also want to verify that the initial device orientation and autorotation match those of the Cocos2D project.

FirstSceneClassName is a class name, not file name!
Note that this parameter should indicate the class name, not the file name. In the case of the particles project the file names of the first scene was HelloWorldScene.h respectively .m but the class was actually called just **HelloWorld**.

Do not remove entries from **config.lua** which are seemingly unneeded. Kobold2D may rely on those, and you may later want to be able to change some of these settings without having to look them up in the Kobold2D documentation.

If you are unsure which settings were used by your Cocos2D project, have a look at that project's AppDelegate.m and GameConfig.h files. If you don't remember making any changes to these files the defaults provided by Kobold2D should work fine. If you do encounter strange issues in the Kobold2D version of your project, such as multi-touch not working or performance degradation, it is most likely caused by different startup settings. Please refer to the Config.lua Settings Reference for more information.

Changes to your "first scene" class



If you allow Kobold2D to run your first scene via the config.lua setting **FirstSceneClassName** then the standard class method provided by Cocos2D project templates, namely the method `+(id) scene` is no longer being called. If the scene method contains just the default code adding a layer to the scene, you will probably not notice a difference. But if you have custom code in that method, you have these options:

1. rename the method from `(id) scene` to (id) node in the implementation and interface
2. you can also move the `+(id) scene` code to layer's init method
3. call your first scene manually in the AppDelegate method `-(void) initializationComplete`

Option #1 is the simplest solution. It should work in most cases flawlessly. But if not you might want to try the other options.

Option #2 is usually trivial, if all you did in the scene method was adding more layers. Those layers can also be added in the init method of the same class. If that class is itself a layer, try adding the new layers to `[self.parent addChild:myLayer]` instead of directly to self (the HelloWorld class in this case).

Option #3 is also trivial and simply overrides Kobold2D's way of running the first scene by reverting to the old-school, Cocos2D way to launch the first scene. The code added to the AppDelegate class would look similar to this and ensures that the `+(id) scene` method is called as usual:

```
-(void) initializationComplete
{
 [[CCDirector sharedDirector] runWithScene:[MyFirstScene scene]];
}
```

Other Alterations

Some projects may require additional changes or manually migrating code. That is where you should carefully test each step in the "migratory process" and where we'll find it difficult to provide support, as every project is different. But feel free to ask for help in the Kobold2D Forum.

Take Small Steps
If you have to tread unpaved territory when migrating your project it helps to first get your project up and running, if incomplete. Then migrate custom code step-by-step in small bunches to the Kobold2D project, if necessary commenting out some code to make it work. Always test each step. Repeat this process to slowly but surely and consciously get closer to the final, working project.

Migrating AppDelegate & RootViewController changes

In particular if you have modified the AppDelegate or RootViewController classes you will have to migrate these changes step by step to the Kobold2D project. Just be sure to call the super implementations for any overridden methods in those classes, as Kobold2D implements default AppDelegate and RootViewController classes which perform default operations, such as properly enabling and performing autorotation based on the config.lua settings.

Migrating Default Resources

Remember the non-mental notes you should keep? Now is the time to compare the Info.plist files of the two projects and merging any changes to the ones in the Kobold2D project.

Likewise, any modified default image (Default.png, Icon.png) you should now simply copy over the existing ones in the Kobold2D project's Resources folders, using Finder. You can then verify in Xcode by selecting these files that they have been correctly updated.

Migrating a third party library not provided by Kobold2D

This depends on how you've added that library to your project. If it was, like Cocos2D, simply added to your project by dropping the library's source code files to your project's target, you can just treat that library's code as if it were your own.

If the code is compiled as a static library you may have to add that particular target (or one for each supported platform) to your Kobold2D project.

Let us know!
If you think the library you're using would be great to have in Kobold2D, and provided that library is free to distribute and open source, you might want to request that library being added to Kobold2D. You can do so in the Kobold2D Forum.

Warnings and errors that weren't there before

Migrating code to a Kobold2D project can give you warnings or errors that weren't there before. It's possible that the code still works but isn't 100% conforming to good coding practices. For example, the Xcode default build settings will not warn you about possibly undeclared selectors, instead your app will just crash while it is running. Kobold2D has this warning enabled by default, and also turns all warnings into errors since warnings should be taken serious.

The simplest fix is to find the offending compiler warning setting and to turn it off (or on) in your project's build settings. But you might want to take the opportunity and check if the new warnings/errors indicate potential bugs in your project.

Removing unsupported platforms

If your project does not support a particular platform, be it iOS or Mac, then you may want to remove:

- the platform's target
- the platform's build scheme
- platform-specific resource files (see Projectfiles-iOS respectively Projectfiles-Mac)

You may want to consider keeping the unnecessary platform specific targets if there's even a slight possibility that you might want to go cross-platform eventually. Adding a platform specific target back in isn't as straightforward as it may seem.

The Kobold2D Template Projects

Learn iOS Game Development

The Learn Cocos2D book symbol indicates in which chapter or article you can learn more about each template project.

Basic Template Projects

Hello Kobold2D

iOS

Mac OS

The standard template project for Kobold2D is the successor of the "Hello Cocos2D" template. It showcases Retina support, config.lua settings, OS and device detection, iAd banners, actions, built-in and custom font labels and sound.

Chapter 16

Use this template as a starting point if you want to start a Kobold2D project from scratch.

Hello Cocos3D

iOS

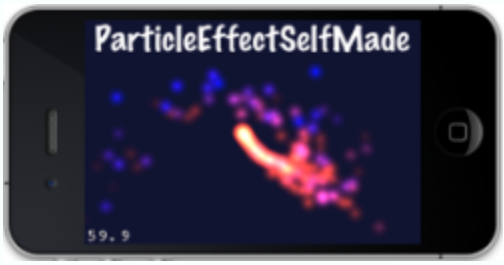
iOS

This template is similar to *Hello Kobold2D* and showcases the use of Cocos3D, mixed with regular Cocos2D nodes. Notice that Cocos2D nodes can be drawn in front of or behind 3D objects. Cocos3D currently does not support Mac builds.

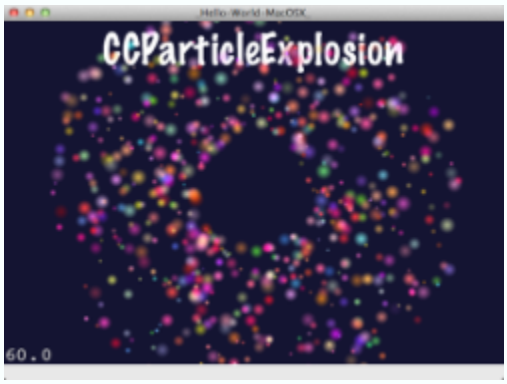
Chapter 16

Use this template as a starting point if you plan to work with Cocos3D. It has the required line `-force_load libcocos3d.a` in `BuildSettings-iOS.xcconfig` already set.

Particle Effects



iOS



Mac OS

This project shows you how to create particle effects by subclassing the CCParticleSystem class. It also displayed particle effects created with Particle Designer as well as Cocos2D's built-in particle effects.

Chapter 9

User Input

iOS

Mac OS

Illustrates how to use the KKInput class which wraps user input into a collection of convenience method. The main difference being that KKInput allows you to poll the input state at any time from any class without having to register to receive input events. KKInput takes the grunt work out of input processing, giving you the ability to easily test for double-clicks, scroll wheel & mouse position changes, key states with or without modifiers, and more.

Exclusive Kobold2D Feature!

Advanced Template Projects

Physics Box2D / Physics Chipmunk / Physics Chipmunk SpaceManager

iOS

Mac OS

The three physics template projects in Kobold2D all implement the same physics demo: adding dynamic bodies (boxes), collision callbacks, and the screen/window acting as an impenetrable barrier. A static body is used to attach three dynamic bodies which are connected via Revolute (Box2D) / Pivot (Chipmunk) joints to create a swinging "rope".

Chapter 12

If you're unsure which physics engine is right for you, check out these templates and use the one which you find easiest to work with. You might also want to read the FAQ entry: Which Physics Engine is the best?

Orthogonal Tilemap

iOS

Mac OS

The orthogonal tilemap project shows how to load a TMX tilemap created with the Tiled Map Editor. It demonstrates conversion of touches to tile coordinates, inspecting tile properties, the use of the Tiled Object Layer for trigger areas and scrolling of the tilemap.

Chapter 10

Isometric Tilemap

iOS

Mac OS

The isometric tilemap project shows how to load an isometric TMX tilemap created with the Tiled Map Editor. The isometric tilemap is drawn with depth buffering and 2D projection for correct Z ordering of tiles. The project demonstrates conversion of touches to tile coordinates, inspecting tile properties, the use of the Tiled Object Layer for trigger areas and scrolling of the tilemap. It includes a controllable player character. On Mac OS it illustrates how to process keyboard events.

Chapter 11

Sprite Performance

iOS

Mac OS

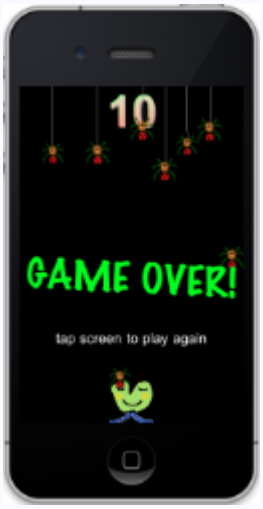
This project illustrates how to improve rendering performance by replacing sprites created from individual files to sprites created from sprite frame provided by a PVR compressed texture atlas. It also shows how to use CCSpriteBatchNode to improve performance even further.

This project can also be used for benchmarking sprite rendering performance of various sizes and image formats.

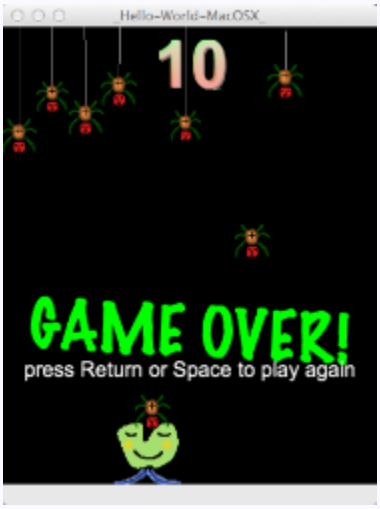
Blog Post

Game Template Projects

Doodle Drop



iOS



Mac OS

Doodle Drop is a fully playable game in which you attempt to avoid the dropping spiders by tilting your device. The game showcases accelerometer input (keyboard input on Mac), sound, music, a game over screen, a custom loading screen, score counting wth a bitmap font label, object pooling to avoid frequent alloc/dealloc cycles and advanced use of easing, rendering lines, simple collision testing, sequence and call func actions.

Chapter 4

Parallax Side-Scroller

iOS

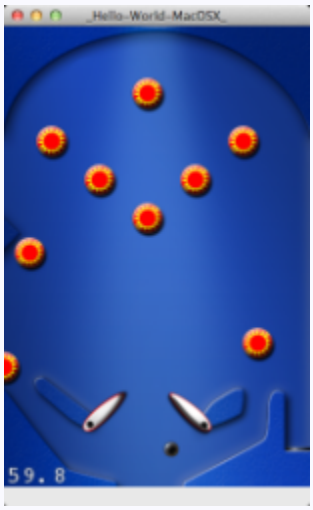
(no Mac OS version)

The Parallax Side-Scroller is a classic shoot'em up game with a virtual joypad and button provided by SneakyInput. The project shows you how to create an endlessly scrolling parallax background, how to structure bigger projects into multiple classes, how to pool and re-use game objects, how to keep nodes within a defined area by overriding setPosition, how to create reusable game components, how to turn your game's scene into a Singleton to make it accessible to other classes, and how to access other classes through your game's scene.

Chapters 6-8

Pinball Game

iOS



Mac OS

The Pinball project is a functional pinball table with bumpers, flippers, a plunger and a ball. Box2D is used for physics simulation, the collision shapes were created with PhysicsEditor. The plunger and flippers are animated via Box2D motors. Collision events are forwarded via Objective-C messages to individual objects and correspondingly named selectors (eg beginContactWithBall, endContactWithPlunger). Includes a demonstration of gravitational pull by uncommenting the function applyForceTowardsFinger.

Chapter 13

Creating a Kobold2D Template Project

You can create your own template projects with ease. Template projects are projects that you use as basis for new projects, with more or less boilerplate code already included. Template projects will be listed in the Kobold2D Project Starter tool as described in Kobold2D.

Reasons for creating your own Template Projects?
You may want to share your own template projects with the community, maybe with an accompanying tutorial. This is a sure way to get attention.

Businesses can use template projects to ensure that new projects include all the shared source code, assets, build schemes, etc. that you have developed over time, or need to have in every project (eg. OpenFeint).

You can also use template projects as starting points for the customers of your commercial add-on library for Kobold2D.

Before you begin ...

It is highly recommended to build a template project based on one of the already existing template projects provided by Kobold2D. After all, you want to end up with a working Kobold2D project.

If in doubt ...
... check how the existing Kobold2D project templates are setup. You can open the template projects in the `/__Kobold2D__/templates/project/` folder and review the project, but it won't build. To be sure you don't accidentally break the template project, copy it elsewhere before opening the .xcodeproj file.

Correctly Naming a Template Project

Template projects rely on a common naming scheme for the project's folder, Xcode project, targets and schemes. The naming scheme is as follows:

_ + The-Project-Name + -Template_

For example if your template project's name is supposed to be "Hello My Friend" then the project name will be `_Hello-My-Friend-Template_` . The rest of this guide assumes that `_Hello-My-Friend-Template_` will be the chosen name for the template, but of course you can change the `Hello-My-Friend` part to whatever you like.

It is recommended to replace all spaces with dashes, and to avoid using any punctuation or other special characters. Template Projects and the Kobold2D Project Starter tool have only been tested with names containing underscores, dashes, letters and digits.

When you create a new project by using the Kobold2D Project Starter tool, all occurances of the string `_Hello-My-Friend-Template_` will be replaced by the project name the user chooses.

This replacement is done for the project's folder, the .xcodeproj file, the project's targets, schemes and groups. This replacement however does **not** extend to source code or resource files.

To summarize, make sure that the following is true in your template project:

- the project and its files are in a folder named: `_Hello-My-Friend-Template_`
- the project file (.xcodeproj) is in that folder and named: `_Hello-My-Friend-Template_.xcodeproj`
- the project's targets are named: `_Hello-My-Friend-Template_-iOS` for iOS targets and `_Hello-My-Friend-Template_-Mac` for Mac targets. The `-iOS` and `-Mac` suffixes are optional but help to identify the target platform.
- the project's schemes are named exactly like the project's targets: `_Hello-My-Friend-Template_-iOS` and `_Hello-My-Friend-Template_-Mac`

If your template project only supports one platform, simply omit the target and scheme for the platform(s) that the project doesn't support.

Adding a Description

To add a description that will be displayed by the Kobold2D Project Starter tool, simply add a text file named `_description_` in the same folder as the .xcodeproj file. Verify that the description text fits into the description box of the Kobold2D Project Starter tool.

Moving the Template Project

After you've finished renaming your template project, along with all other work, you will have to move or copy the project's folder to the project templates folder `/__Kobold2D__/templates/project/`.

Now you can run the Kobold2D Project Starter app. Your template project should show up. Select it and create a new project based on your template project, preferably give it a name that stands out (eg. `TEMPLATETEST`) because that will make the next step easier.

Verify that the project folder, Xcode project, targets and schemes have been properly renamed to `TEMPLATETEST` or whatever name you chose. The targets and schemes should be named `TEMPLATETEST-iOS` and `TEMPLATETEST-Mac`.

Building and Running the Template Project

If the names have all been correct replaced, verify that the `TEMPLATETEST` project builds without errors.

If the `TEMPLATETEST` project builds successful it will be sufficient to test future changes to your template project in the template project *`Hello-My-Friend-Template`* itself. You only occasionally will have to verify that the project also builds without errors after going through the Kobold2D Project Starter tool.

Open the **Product** menu in Xcode, hold down the Option key and select **Clean Build Folder ...** to make sure everything gets build anew. You only have to do this step once before building any of your newly created project's targets.

You will want to build each target (iOS & Mac) once, and run it to see that it behaves as it should. You should build and run the iOS target once for each of the possible selections: iOS Device, iPhone Simulator and iPad Simulator. Even if your template does not support iPad its good practice to make sure it at least runs on iPad.

You also will want to verify that the project behaves correctly on both standard and Retina displays. Run the project once using the iPhone Simulator as target. Verify the app is correct, then in the Simulator go to the **Hardware -> Device** menu and switch to the other iPhone type, which will be either **iPhone** or **iPhone (Retina)**. Then hit the run button in Xcode again, or simply tap the app's icon in the Simulator. This way you can quickly ensure that your app displays correctly on standard and Retina devices.

Finally, you should build and run the Mac targets for both 64-Bit and 32-Bit modes, assuming your installation of Mac OS X supports both. This is

important because sometimes compile or runtime errors will only occur in either 32-bit or 64-bit builds.

> **One more reason to share template projects...**
> ... the best, most popular template projects created by the community will be included in Kobold2D if the author(s) allow it to be distributed under the MIT License.

You're done!

You can now share the template project with your team or the entire community. Right-Click the project's `_Hello-My-Friend-Template_` folder and select *Compress "_Hello-My-Friend-Template_}}. You can rename the archive to whatever you like.

> Provide your users with installation help by linking to this article:
> Installing a Kobold2D Template Project

Installing a Kobold2D Template Project

> If you receive an archive containing a Template Project for Kobold2D, this quick guide explains how to install it.
>
> Developers of template projects should include the following link along with the template download link (on your webpage, forum post, or readme file) as instruction for users of their template project:
> Installing a Kobold2D Template Project — `http://www.kobold2d.com/x/ZwUO`

After downloading and extracting an archive containing a Kobold2D template project you should notice a new folder named similar to this example: `_Hello-My-Friend-Template_`

Copy or move the folder to the Kobold2D templates folder. By default the templates folder is located here:

```
~/Kobold2D/Kobold2D-x.y/__Kobold2D__/templates/project/
```

Whereas Kobold2D-x.y will be the versioned Kobold2D folder, for example: `Kobold2D-1.0`

> If the project template was created for an older version of Kobold2D and hasn't been maintained, it's probable that projects based on this template project may not compile, contain bugs, or won't even show up in the Kobold2D Project Starter tool.

After copying the folder you can launch the Kobold2D Project Starter tool and start a new project based on the newly installed project template.

> If the template project does not show up, and in particular the template's folder does not begin with {} *(underscore) and end in* `-Template` please contact the project template author for support.

Kobold2D

Online Reference Manuals

Cocos2D

Cocos3D (:)

CocosDenshion FAQ
CocosDenshion CookBook
ObjectAL in a Nutshell

Box2D User Manual
Chipmunk Documentation

Lua Reference Manual

Online API References

- Box2D API Reference
- Chipmunk API Reference
- Chipmunk SpaceManager API Reference
- Cocos2D API Reference (iOS)
- Cocos2D API Reference (Mac OS)
- Cocos2D Extensions API Reference (iOS)
- Cocos2D Extensions API Reference (Mac OS)
- Cocos3D API Reference (iOS)
- CocosDenshion API Reference (iOS)
- CocosDenshion API Reference (Mac OS)
- Kobold2D API Reference
- ObjectAL API Reference (iOS)

- SneakyInput API Reference

API References

Get help in Xcode!
Kobold2D installs the API References into Xcode's help system. In Xcode, choose **Help -> Xcode Help**, then switch to the **Editor -> Explore Documentation** panel.

Box2D API Reference

Open in new window ...

Chipmunk API Reference

Open in new window ...

Chipmunk SpaceManager API Reference

Open in new window ...

Cocos2D API Reference (iOS)

Open in new window ...

Cocos2D API Reference (Mac OS)

Open in new window ...

Cocos2D Extensions API Reference (iOS)

Open in new window ...

Cocos2D Extensions API Reference (Mac OS)

Open in new window ...

Cocos3D API Reference (iOS)

Open in new window ...

CocosDenshion API Reference (iOS)

Open in new window ...

CocosDenshion API Reference (Mac OS)

Open in new window ...

Kobold2D API Reference

Open in new window ...

ObjectAL API Reference (iOS)

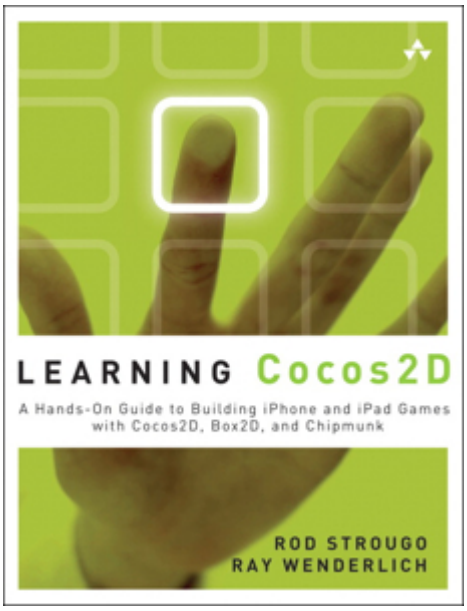
Open in new window ...

SneakyInput API Reference

Open in new window ...

Cocos2D Resources

Learn iPhone and iPad Cocos2D Game Development Learning Cocos2D



Authors:	Steffen Itterheim, Andreas Löw
Published:	November 2010 (1st Edition) October 2011 (2nd Edition)
Cocos2D Version:	v0.99 (1st Ed) — v1.0 (2nd Ed)
Source Code:	Free Download: v0.99 — v1.0
Websites:	Book Website Apress Website

Authors:	Rod Strougo, Ray Wenderlich
Published:	July 2011 (1st Edition)
Cocos2D Version:	v1.0 (1st Ed)
Source Code:	Free Download: v1.0
Websites:	Book Website Addison-Wesley Website

Why are these books recommended?
The above books are up to date, accurate, easy to follow and cover a broad scope of topics — including UIKit integration, physics engines, tilemaps, performance optimizations and development tools.

Cocos2D Tutorials

There are plenty of tutorials for Cocos2D on the net. But only a few tutorial authors stand out from the crowd:

- Ray Wenderlich — Cocos2D & iOS Tutorials

- Mohammad Azam — Cocos2D & Mobile Development Tutorials and Videos
- SDKTutor — Cocos2D Tutorial Videos

Cocos2D Podcast

The Cocos2D Podcast is hosted by Mohammad Azam and Steffen Itterheim. They're often joined by guest speakers from the community to talk about Cocos2D, Game Development, and other things of interest.

The Cocos2D Podcast is also available on iTunes.

Cocos2D Wiki

Then there's the official Cocos2D Documentation. You can learn a few bits and pieces from it but unfortunately it doesn't teach you Cocos2D in its entirety. That's a big blank which the books above filled in.

Kobold2D Reference Manual

Config.lua Settings Reference

The KKStartupSettings in the **config.lua** file included in each Kobold2D project are described in detail in the Kobold2D API Reference: KKStartupSettings class reference

Note that the KKStartupSettings properties begin with a lowercase character which must be written as uppercase in the config.lua file:

KKStartupSettings Property name:	config.lua Parameter name:
gLViewColorFormat	GLViewColorFormat
deviceOrientation	DeviceOrientation
enableMultiTouch	EnableMultiTouch

Example config.lua

```
local config =
{
 KKStartupConfig =
 {
  -- Load first scene from a class with this name
  -- The class must inherit from CCScene or CCLayer.
  FirstSceneClassName = "GameLayer",

  -- set the director type, and the fallback
  DirectorType = DirectorType.DisplayLink,
  DirectorTypeFallback = DirectorType.NSTimer,

  MaxFrameRate = 60,
  DisplayFPS = NO,

  EnableUserInteraction = YES,
  EnableMultiTouch = NO,

  -- Render settings
  DefaultTexturePixelFormat = TexturePixelFormat.RGBA8888,
  GLViewColorFormat = GLViewColorFormat.RGB565,
  GLViewDepthFormat = GLViewDepthFormat.DepthNone,
  GLViewMultiSampling = NO,
  GLViewNumberOfSamples = 0,

  Enable2DProjection = NO,
  EnableRetinaDisplaySupport = NO,

  -- Orientation & Autorotation
  DeviceOrientation = DeviceOrientation.Portrait,
  AutorotationType = Autorotation.CCDirector,
  ShouldAutorotateToLandscapeOrientations = NO,
  ShouldAutorotateToPortraitOrientations = YES,
  AllowAutorotateOnFirstAndSecondGenerationDevices = YES,

  -- iAd setup
  -- Note: for iAd to support autorotation you should use:
  -- AutorotationType = Autorotation.kAutorotationUIViewController
  EnableAdBanner = NO,
  LoadOnlyPortraitBanners = NO,
  LoadOnlyLandscapeBanners = NO,
  PlaceBannerOnBottom = NO,

  -- Mac OS specific settings
  AutoScale = YES,
  AcceptsMouseMovedEvents = NO,
  WindowFrame = RectMake(1024-640, 768-480, 640, 480),
 },

 -- you can create your own config sections
 HelloWorldSettings =
 {
  HelloWorldString = "Hello Kobold2D!",
  HelloWorldFontName = "Marker Felt",
  HelloWorldFontSize = 50,
 },
}

return config
```

KKStartupConfig Reference

Could not retrieve http://www.learn-cocos2d.com/api-ref/latest/Kobold2D/html/interface_k_k_startup_config.html - Page Not Found
Could not retrieve RSS feed: no URL

Lua Reference Manual

Open in new window ...

Kobold2D FAQ

Kobold2D General FAQ

Kobold2D Xcode FAQ

Kobold2D Technical FAQ

Kobold2D Cocos2D FAQ

Kobold2D Wax & Lua FAQ
• Can I write game logic in Lua with Kobold2D? • What's the difference between Wax and Corona SDK?

Kobold2D Audio FAQ
• Which audio engine is better - CocosDenshion or ObjectAL?

Kobold2D Physics FAQ
• Which Physics Engine is the best?

Kobold2D Cocos3D FAQ
• Is there a Mac OS version of Cocos3D?

Kobold2D General FAQ

Why was Kobold2D created?

Oh, so many reasons...

Very high up on the list of reasons is a general dissatisfaction with the development of cocos2d-iphone beginning in early 2010. That and a need for a more professional work environment triggered the development of conceptual features that ended up forming the backbone of Kobold2D, such as an easy upgrade path and including preconfigured libraries.

When was Kobold2D created?

Kobold2D was first published in August 2011. Some of the concepts implemented in Kobold2D date back to early 2010.

What does Kobold2D cost?

Nothing.

Kobold2D is free and open source. It's released under the MIT License.

We do sell commercial add-on products and affiliate products to support the development of Kobold2D.

Can I donate to Kobold2D?

No.

First of all: donations bring in very little money. It's like trying to convince a software pirate to buy the software he just downloaded for free. Very, very few do it. Plus everyone thinks there are enough others doing it. So, there you go.

Commercial products are much more effective to cover costs and to subsidize the development of free products. And the users actually gets something valuable in return. Win-win.

In addition, it simply wouldn't be right to sell commercial products and then also accepting donations at the same time. At least not without being fully transparent about what the donations are being used for.

There's also the question of fairness: if the donations do cover all of the running costs, how is the extra money used? How should we split donations among contributors and library authors fairly? How do we provide enough transparency about how high the running costs are and how much of it has been covered through donations?

Simply put: we'd rather have you donate to other library authors who need donations.

Why does Kobold2D consume 400 MB disk space?

Currently, the Kobold2D download is about 140 MB and the size of Kobold2D installed on your computer is over 400 MB. Each new version of Kobold2D will add another 330 MB or more to your hard drive.

Kobold2D includes several libraries and many project templates. But the biggest size factor is the documentation in the `./_Kobold2D_/docs/` folder, around 200 MB. Over 80 MB are consumed by the Xcode docset help files which are copied to the user's Docset folder to make them available in Xcode.

New Kobold2D versions are installed to new folders (eg `Kobold2D-1.0.0`, `Kobold2D-1.0.1` , and so on) to enable migrating projects to newer Kobold2D versions with the help of the Kobold2D Project Upgrade tool.

See also: How can I reduce the installed size of Kobold2D?

How can I reduce the installed size of Kobold2D?

You can safely delete the .docset files in the `./_Kobold2D_/docs/` folder. They're already copied to `~/Library/Developer/Shared/Documentation/DocSets/`. You can also delete the docset files in the latter directory, if you don't use the Xcode help for Kobold2D libraries at all.

If you do not need the (somewhat faster) Kobold2D offline documentation you can safely delete the entire `docs` folder. In particular you might consider doing this with older versions of Kobold2D that you don't actively work with anymore, but want to keep around because a project built with this particular Kobold2D version is already in the App Store. You never know when that project might need a bugfix or compatibility update for a new iOS version.

The Kobold2D Online Documentation allows you to access older versions of the documentation.

Of course, you can also delete old Kobold2D versions entirely once you have migrated all of your projects to newer Kobold2D versions. Do this by deleting the folder with the version number suffix, for example:
`Kobold2D-1.0.0-preview`

How can I install Kobold2D to a custom folder?

You can move the `~/Kobold2D` folder after installation to wherever you want.

There's only one thing to consider: for the Kobold2D Project Upgrader tool to work every versioned Kobold2D folder (eg. `/Kobold2D-1.0`) should be in the same folder, so that the folder structure looks something like this:

```
/MyProjects/KoboldProjects/Kobold2D-1.0
/MyProjects/KoboldProjects/Kobold2D-1.1
/MyProjects/KoboldProjects/Kobold2D-1.2
/MyProjects/KoboldProjects/Kobold2D-1.3
```

Otherwise the Kobold2D Project Upgrader tool won't be able to find upgradeable Kobold2D Projects.

The Kobold2D installer doesn't allow you to install Kobold2D to a different folder. It will always install Kobold2D to `~/Kobold2D`. This is done so that users without a system administrator account can install Kobold2D.

How can I uninstall Kobold2D?

It's simple: delete the folder `~/Kobold2D` or a particular versioned folder, for example: `~/Kobold2D/Kobold2D-1.0`

> ⚠ This will also delete any Kobold2D projects that you have created.

If you want to uninstall in order to re-install Kobold2D, you can simply run the installer again. It will overwrite any Kobold2D files that already exist but none of the files you created.

Kobold2D Xcode FAQ

Why does Xcode crash after installing Kobold2D or the Xcode Help files?

> ⚠ **Close Xcode during installation**
> It is recommended to quit Xcode before installing Kobold2D.

Xcode may crash if it is running while performing one of these actions:

- installing a new version of Kobold2D
- re-installing the same version of Kobold2D
- running the `/_Kobold2D_/docs/install-all-docsets.sh` script

The cause of the crash is that Xcode Help does not like existing and open docsets (Xcode help files) to be removed or replaced. This is what happens in the above cases and could cause Xcode to report an error, after which you are prompted to either "Ignore" the error (which may not work) or "Crash" Xcode.

How can I create a blank Kobold2D project?

Use the **Kobold2D Project Starter.app** as described in Kobold2D and then modify the newly created project.

> ⚠ **Only Kobold2D Project Starter.app created project are supported!**
> While you can create and add any project to the Kobold2D workspace, such projects will not benefit from the Kobold2D features. The same goes for existing non-Kobold2D projects being added to the Kobold2D workspace.
>
> The process of correctly creating a working Kobold2D project is very involved and not documented. It is a lot easier to simply modify one of the provided template projects. If you want to migrate an existing project to Kobold2D, you would also start by first creating a new Kobold2D project.

Can I add the Kobold2D.xcodeproj to my project?

That's not supported.

You can try and add the Kobold2D.xcodeproj to an existing project, but it won't magically turn your existing project into a Kobold2D-enabled project. More likely you'll get compile errors that we'll be unable to help you with.

Likewise, just adding your project to the Kobold2D.xcworkspace will not make your project a Kobold2D-enabled project.

Instead, this question ultimately seems to be about Migrating a Cocos2D Project.

Is Kobold2D compatible with Xcode 3?

No.

Kobold2D uses a workspace to combine multiple projects into a single workspace. This, among other improvements that Kobold2D makes use of, is only available in Xcode 4.

You can have both Xcode 3 and Xcode 4 installed on your system. Just install each SDK to a different folder on your hard drive. Instead of the default `/Developer` you could install to `/Developer-XC3` and `/Developer-XC4`.

Why does Kobold2D build successfully but won't run?

If Kobold2D builds successfully but won't open the iOS Simulator, a Mac OS window, or deploy to the device, you may not have created a Kobold2D project with the Kobold2D Project Starter tool after installation.

Adding a Project

If you open the Kobold2D.xcworkspace, and it only contain a single project named `Kobold2D-Libraries.xcodeproj` (contains Kobold2D's static library projects) then there's nothing that *can* be run.

Kobold2D explains how to create a Kobold2D project that you can run.

Selecting the Project Scheme

If you do have a custom project in the workspace, you may not have the build scheme for that project selected in Xcode. Use the drop-down list on the upper left area of Xcode to select the scheme for your project.

Can I work in a copy of the Kobold2D Workspace?

Yes, as long as the copied xcworkspace resides in the same folder as the `Kobold2D.xcworkspace`.

The Kobold2D Project Starter tool allows you to create new projects in a custom workspace simple by choosing an existing workspace, or entering the name of a new workspace.

Can Kobold2D Projects be located in a custom folder?

Theoretically yes, but it's unsupported and strongly discouraged.

Every new project you create with the Kobold2D Project Starter app will reside in the versioned Kobold2D folder (eg `/Kobold2D-1.0`). The project will either be added to the Kobold2D.xcworkspace or reside in its own workspace. Neither the workspace nor the project should be moved to a different folder, it will stop working if you do so.

Most importantly, if you move your workspace and project outside the versioned Kobold2D folder, the Kobold2D Project Upgrader tool will be unable to upgrade that project to a newer Kobold2D version. Other tools may also stop working.

Possible Solution for Power Users (Unsupported)

Voiding Warranty
The following steps are for those who feel confident in changing the Kobold2D project structure. We strongly recommend that you do not do this, and we will not give support for the following modifications.

Please ask yourself what reasons you have for changing the Kobold2D project structure. We programmers often like to have things our own way and customize our work environment because it just feels right. But often we do so without even trying to solve very particular issues and without consideration for all the potential drawbacks of that customization.

The "still ok" solution to custom folders

Kobold2D workspaces rely on the `Kobold2D-Libraries.xcodeproj` to be in the relative path `/__Kobold2D__/Kobold2D-Libraries.xcodeproj`. If you move your workspace to a custom folder it obviously won't be able to find the `Kobold2D-Libraries.xcodeproj` anymore. While you can simply change the path to the `Kobold2D-Libraries.xcodeproj` you should ask yourself if this is really what you want.

The much better alternative would simply be to copy the `__Kobold2D__` folder along with your project and workspace, and so your workspace continues to work and your project is tied to a very particular version of Kobold2D.

If you keep all of your custom folders at the same level even the Kobold2D Project Upgrader tool would still work.

The "inconvenient but workable" solution to custom folders

If you still decide to keep your workspace and project in a different folder from the __`Kobold2D`__ folder, be aware of the following issues:

For one, you'll have to dig deep in your workspace to find out with which `Kobold2D-Library.xcodeproj` your workspace is currently linked with. It could be the one in the `/Kobold2D-1.0` folder or it could be the one in the `/Kobold2D-1.1` folder, or some other version.

Without actually looking it up in the workspace you won't know. And you will have to maintain that connection manually in your workspace. You should also avoid renaming or moving the Kobold2D versioned folders because you'll never know which of your custom-located workspaces is linked with that particular `Kobold2D-Library.xcodeproj`.

It's an inconvenient but workable solution.

The "thing that should not be" ~~solution~~ recipe for disaster to custom folders

If your intention is to keep only a single version of Kobold2D and its libraries in a commonly named folder (eg simply `/Kobold2D`) then every time you upgrade that Kobold2D version, all of your custom workspaces and projects will be using the latest Kobold2D version automatically. That may seem like a good idea at first, but it's really not. It's the worst possible setup you can have.

Consider this: You've built an app with Kobold2D-1.0 that's been in the App Store for 6 months. The project remained untouched ever since. Now you have to update that App due to an iOS update and an incompatibility caused by it. You open the project only to find out that it doesn't build anymore because in the meantime you've upgraded Kobold2D to v2.0. Instead of a quick fix you'll be forced to adapt your entire project to a new API and re-test everything.

Now you'd think you could just downgrade the Kobold2D version temporarily. But alas, you don't even know for sure which Kobold2D version your project was built with. It could have been v1.0 but it could have also been v1.1 or v1.2.

If you work with source control and label your releases, including all referenced libraries, this won't be that much of a problem for you. But it's still inconvenient and there's the potential for human error (forgot to label a release) that may reveal itself only months later.

Will installing Kobold2D affect my Cocos2D Projects?

No, not in any way.

Your Cocos2D Projects remain completely separate from the Kobold2D installation. Kobold2D doesn't overwrite or otherwise modify Cocos2D or its Xcode templates.

But maybe you are interested in Migrating a Cocos2D Project? You can do so safely. If you follow the migration guide your Cocos2D project also remains unchanged.

How can I copy the Build Log in Xcode?

For reporting compile-time issues with Kobold2D you may be asked to post the full build log. Here's how you can copy the build log in Xcode 4:

1. In Xcode, open the **View** menu and choose **Navigators -> Log Navigator** (hotkey: Command + 7).
2. Select the topmost item in the Log Navigator list that starts with "Build".
3. The build log is to the right of the Log Navigator. Click on any log entry just to change the focus to the build log panel.
4. Then choose **Edit -> Select All** (hotkey: Command + A) followed by **Edit -> Copy** (hotkey: Command + C). This copies the entire build log as text to the clipboard
5. You can then paste the build log anywhere.

The log can be quite long, if it is too long to post in the forum, save the log as text (eg with TextEdit.app) and attach the txt file to the post or email it.

Does Kobold2D work with the latest beta SDK?

Maybe, maybe not.

No beta SDK Support!
Kobold2D generally doesn't support Apple iOS SDK beta or Mac OS X SDK beta builds. Any incompatibilities will not get looked at until Apple has released a GM Seed or final version of the latest SDK.

Kobold2D is somewhat at the mercy of other libraries. For example, cocos2d-iphone also fixes incompatibilities only after the new SDK has been officially released.

General Tips About Apple's Beta SDKs

Do not talk about beta SDKs!
Rule #1: Do not talk about Apple beta SDKs!
Rule #2: Do not talk about Apple beta SDKs!

There's a reason for that: beta SDKs are under NDA (non-disclosure). You are not allowed to publicly talk about beta SDKs. Once you allow users to discuss issues that appear due to Apple's beta SDKs the likelihood that someone mentions something that is under NDA is high. Similarly support is hard and sometimes impossible if you can't mention the very thing that you need to talk about to explain a possible solution.

Always install beta SDKs separately!
If you have to work with a beta SDK, make sure that you install it to a different directory and not `/Developer`. You might want to suffix the `/Developer` folder with the version number of the beta SDK, for example `/Developer51` if you're installing the preview version of iOS SDK 5.1.

You can then switch between the latest official SDK and the beta SDK by running the tools (eg Xcode) from the corresponding folder.

Do you really need the beta SDK?
Unless you **must** develop at the bleeding edge, you should prefer to wait for the public release of the new SDK. You generally don't get support for beta SDKs anywhere but the closed Apple Developer forums, and help for particular libraries and compatibility issues with them is minimal at best.

Only consider installing a beta SDK if:

- you have to use one of the new features in your App.
- you have to test your App very early for compatibility issues, for example because you want your app to be among the first to support a new Apple device.
- you are generally curious and know the risks, know how to install the beta SDK to a separate folder, know how to distinguish between official and beta SDK tools such as Xcode, won't ask questions about the beta version in public.

In any other case it is better to wait for the new SDK to be officially released.

Kobold2D Technical FAQ

Why does Kobold2D compile so much code?

Kobold2D provides all third party libraries as preconfigured, static libraries. By default, all libraries **link to** a Kobold2D project. This causes all the library code to be compiled, whether you use it in your project or not.

Don't worry!
If you concerned about your final app's size, please read this FAQ:
Are Kobold2D apps larger than pure cocos2d-iphone apps?

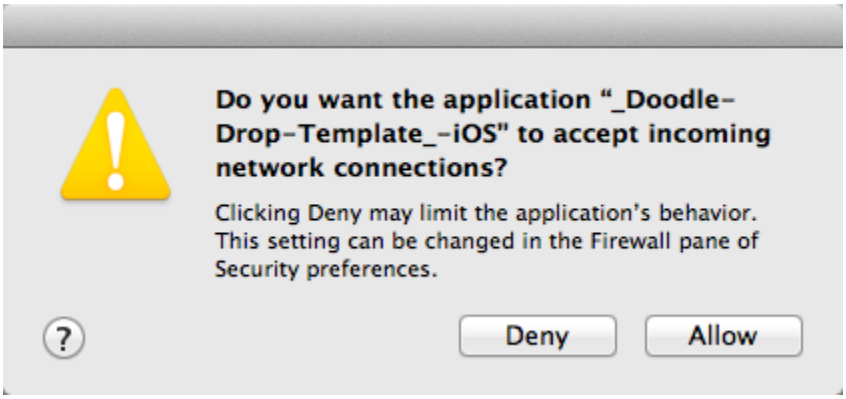
Note that the compilation of the entire Kobold2D code base happens only the first time you compile a new version of Kobold2D, or after switching build configuration (Debug vs Release, iOS vs Mac, iPhone vs iPad vs Simulator), or after a "clean" build.

If you build other projects within the same Kobold2D version, or create a new Kobold2D project, only the project's code will be built since the static library code is already built and available for other projects.

Put simply: from the second build of the same build configuration onwards you're actually saving time!

Where does the 'Network Connections' dialog come from?

When running iOS Simulator builds of a project, you'll see this network connections dialog pop up:



This is nothing to be alarmed of. It is caused by the iSimulate library that allows you to control the action in the Simulator via a Wifi connection from your iOS Device with the iSimulate app installed.

If you don't want or need iSimulate and/or want to get rid of the popup dialog, please read How to disable iSimulate?

Why is the AppDelegate class empty?

The `AppDelegate` class inherits from `KKAppDelegate` which does most of the work for you, including the configuration of Cocos2D and the startup process via the `config.lua` file.

You can still add any `NSApplicationDelegateProtocol` method to the `AppDelegate` class as you see fit. But if you do, you should be sure to call the `super` implementation of that method to allow `KKAppDelegate` to do its job.

Cocos2d-iphone has often caused headaches for developers who updated to a new cocos2d-iphone version but weren't aware that you also needed to make changes to the Cocos2D startup code in your project's app delegate class. Kobold2D avoids this problem by keeping the startup code internal in the `KKAppDelegate` class.

Are Kobold2D apps larger than pure cocos2d-iphone apps?

Typically: no.

Tests with release builds of Kobold2D template project and comparing the results with the cocos2d-iphone project templates showed that the resulting Kobold2D apps are often slightly smaller (several KB) despite the fact that Kobold2D has the entire Lua library built into the app.

Kobold2D does compile more source code but the resulting app size in release builds is typically smaller than projects created with the cocos2d-iphone Xcode templates.

How can Kobold2D apps be smaller than cocos2d-iphone apps?

Kobold2D links the third party code into your app as static libraries. By default, all libraries are added to the **Link Binary with Libraries** build phase. But here comes the clever part: the linker actually knows which libraries are needed and which aren't. Any libraries that you don't use in your app will be discarded.

> The benefit for you is convenience without any drawbacks: If you want to start using a library, it's as simple as adding the correct header file(s) to your source code.

Next up is dead code stripping. The linker is an intelligent beast. Even if you use a library, you'll hardly use all of its functions. Any unused functions are simply removed from the app. There are indications that the linker can do a better job of stripping dead code if that code resides in a

static library.

Finally, several optimizations have been applied to the global build settings of Kobold2D projects, including proper debug symbol stripping in release builds and disabling C++ runtime type information and exception handling, that lead to a smaller executable size.

How to disable iSimulate?

You can (temporarily) disable iSimulate by opening your project's `BuildSettings-iOS.xcconfig` file located in the BuildSettings group. Locate the **FORCE_LOAD_ISIMULATE** line and simply comment it out to disable iSimulate for the time being:

```
// Comment this line if you don't want or need iSimulate.
// Doing so will also remove the "incoming network connection"
// warning dialog in Simulator builds.
FORCE_LOAD_ISIMULATE = -force_load $(KKLIBROOT)/iSimulateSDK/libisimulate-4.x-opengl.a
```

iSimulate is only linked in iOS Simulator builds of your app:

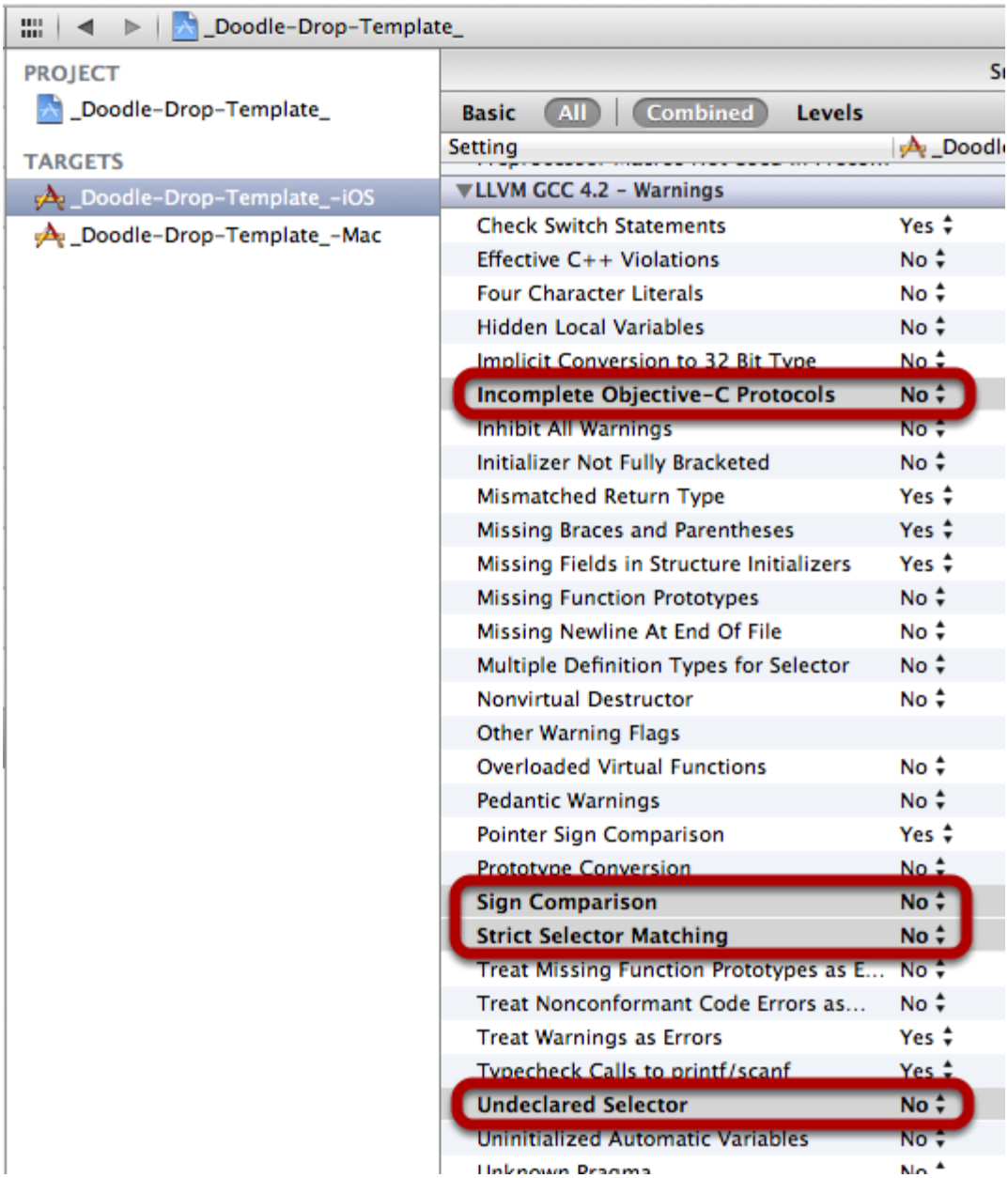
```
OTHER_LDFLAGS[sdk=iphonesimulator*][arch=*] = $(OTHER_LDFLAGS) $(FORCE_LOAD_ISIMULATE)
```

It is not necessary to remove iSimulate from distribution builds.

How to fix compile errors with CocosBuilder?

The CocosBuilder editor for Cocos2D comes with a class called `CCBReader`. If you add the class files to your Kobold2D project it will no longer compile without errors because the CCBReader class doesn't adhere to Kobold2D's stricter warning protocol.

The easiest fix is to disable (set to **NO**) the following warnings in your project's target(s) after adding the CCBReader class:



Kobold2D Cocos2D FAQ

Is Kobold2D compatible with cocos2d-iphone?

Yes.

If you create an empty Kobold2D project and add your existing cocos2d project's source code and resources to it, and assuming the cocos2d-iphone versions are identical, there's no reason why your code should not continue to work.

Is Kobold2D updated when cocos2d-iphone is?

Yes.

As soon as we hear about a new stable release of cocos2d-iphone, we will integrate and test it with Kobold2D. If everything works fine, a new release of Kobold2D will be released within days (hopefully just hours) after the new cocos2d-iphone release.

If any of the other libraries are updated (eg. cocos3d), we normally aim to include the update in the next scheduled Kobold2D update and won't make an immediate release.

Should I try Cocos2D first before switching to Kobold2D?

No.

Kobold2D makes it easier to learn Cocos2D programming! You should learn Cocos2D **with** Kobold2D and skip all the hard parts (like installing templates, setting up a project, etc.). You'll get a lot more templates to get started and 100% working example code to learn from. And so much more.

See also this related question: Do all the Cocos2D tutorials and books work with Kobold2D?

Do all the Cocos2D tutorials and books work with Kobold2D?

Yes, followed by a tiny "but ..."

As far as adding new classes and resources to your project is concerned, you can apply everything you learn from tutorials and books directly to Kobold2D projects.

The "but" part is this: rarely you'll be instructed to make changes to the way Cocos2D is set up. You'll notice this when you're instructed to change code in the app delegate or root view controller classes. In Kobold2D, all those startup settings and function calls have been moved into the `config.lua` script for convenience.

You can either add the code in questions to the app delegate's `-(void) initializationComplete {...}` method, or simply apply the changes to the appropriately named parameters in the `config.lua` file.

Cocos2D version mismatch
If a particular book or tutorial discusses — for example — Cocos2D `v0.99` but the current version is `v1.0` you'll very likely encounter compilation errors simply due to changes in the Cocos2D API (eg renamed classes, removed functionality, etc.). This is neither the fault of Kobold2D nor the fault of the tutorial or book author.

Why is Kobold2D released separately from cocos2d-iphone?

Kobold2D is not and can not be just a contribution to cocos2d-iphone.

The Kobold2D development philosophy and direction are fundamentally different from cocos2d-iphone's. And that's not just about the source code. Spend some time with Kobold2D and you'll be able to spot the differences quite easily.

And we are convinced you'll love them! 

Kobold2D Wax & Lua FAQ

Can I write game logic in Lua with Kobold2D?

In theory yes. But no.

Kobold2D does not provide a Lua scripting interface for game engine classes, eg you can't script game logic in Lua. What you can do is to edit your game's parameters in Lua, then read them in with just one line of code.

Kobold2D relies on Wax which uses the Objective-C runtime to create dynamic bindings to Lua. So you can write Objective-C code in Lua. However, there's really little benefit to that. You lose the ability to debug your code and you'll have to handle awkward syntax that's almost like Objective-C but not quite. The only benefit is automatic memory management, something that Automatic Reference Counting (ARC) will address with iOS 5 anyway.

The Wax binding code through Objective-C runtime features also adds a significant overhead so that scripting games with Wax will give you bad

or terrible performance. To get a good performance you would have to write C functions manipulating the Lua stack directly for each function that you need in your scripting language. This is currently not available in Kobold2D.

I suggest to check out the Corona SDK if you want to write your entire game using Lua with good performance and also be able to deploy to Android devices.

What's the difference between Wax and Corona SDK?

About Corona SDK

The Corona SDK exposes a well designed API in Lua for app and game development. This API is naturally different from the iOS SDK since Corona is also a cross-platform game engine, exposing the same API for all platforms.

For example, when you receive an accelerometer event in Corona you get an event object with these properties. The acceleration parameters are already split into "instant acceleration" and "gravity acceleration", something that iOS developers have to do manually. At the same time, the event also reports whether it was a shake event. On iOS this is an entirely different API from the accelerometer events.

In one sentence, Corona exposes the functionality that the devices offer in a uniform way. Corona also condenses information into fewer properties and methods. You deal with fewer functions which makes it great for rapid prototyping. But sometimes you'll have fewer functionality, and in particular you can't extend Corona with your own Objective-C, C or C++ code since the backend engine is proprietary. Overall, this makes Corona SDK very easy to learn but because it's not extensible and can impose restrictions, it's not for everyone.

About Wax

The iPhone Wax library is a dynamic, automatic binding of the Lua language to Objective-C. Wax provides a translation layer that exposes regular Objective-C code to Lua via Objective-C runtime functions.

Since the binding happens at runtime, it's a **dynamic binding** — in other words there is no code that says that function X should be exposed to Lua in a particular way. Instead, there's code that creates the Lua methods and properties by analyzing the existing Objective-C code at runtime, and applying generic conversion mechanisms to translate between the two languages. The **binding is automatic** because that translation is transparent to the user.

Wax in essence allows you to write Objective-C code in the scripting language Lua. This is nice, in particular automatic memory management should be mentioned as a positive feature of Wax. But other than that it's really hard to justify the use of Wax if you can write the same code in Objective-C. Here are the major drawbacks of writing (scripting) apps with Wax:

1. You end up writing almost the exact same code as in Objective-C. It may be a little less code but it's using a weird and uncommon syntax.
2. The dynamic nature of Wax' Lua binding causes a significant overhead. So much in fact that actual gameplay scripting for realtime (60 fps) games will result in disappointing performance — unless the game is rather simple and/or you target only newer devices (4th generation onward, and iPads).
3. You can not properly debug Lua scripts. You can't set breakpoints, you can't single-step through script code, you can't inspect variables.
4. Lua is a dynamically typed language. That means the compiler won't check Lua scripts for errors. Any syntax errors in your script will only surface while your app is already running.
5. Lua support in Xcode is non-existent. There's no syntax highlighting, no auto-completion, no refactoring, no quick help for Lua script files.
6. You lose named parameters in Lua code. Lua functions are non-descript like C functions: `someFunction("myName", 1, true, "triggerMe", false, 100, 0.4)`

In particular not being able to debug your Wax scripts (#3) and not having compile-time syntax checks (#4) will completely offset any time-savings you might get from writing your app entirely in Wax.

Wax doesn't play into Lua's strengths, which is that it's usually used to expose a simplified, domain-specific API. Such an API would be less error-prone, is easier to understand and debug. Instead, Wax exposes the full complexity of Objective-C and applies a weird syntax, removes debugging, disables syntax checking and reduces code editing comfort. Not on purpose, mind you, but in effect that's what you get.

While Corona shares the issues #4 to #6 with Wax, it's performance and code design is well above that what Wax is able to provide. Corona also comes with a Lua debugger, albeit a command-driven one (eg. like Terminal).

The Solution: A Native Lua Scripting Language

Lua is not without its advantages. It's an excellent language to script games in. But this is only true if the purpose of the language has been determined and an appropriate API has been designed and implemented, exposed to Lua via regular, non-dynamic Lua bindings for optimal performance. That requires a lot of work.

You can bind the exposed API to Lua either by manually writing C functions that manipulate the Lua stack, or via binding libraries such as tolua. Such an approach can have one of these goals:

1. A general purpose game engine API for writing apps in Lua. (Corona SDK)
2. A domain-specific scripting API to manipulate existing game objects at runtime for a specific purpose. This is often tailored to work only with a particular (type of) game. (World of Warcraft UI Scripting)
3. A "load-time" API to configure the app and its objects. Reduces the amount of boilerplate code needed for setting up a scene. Can be used by add-ons and tools as a user-editable data format for scenes, levels, etc. (Kobold2D)
4. Use Number 3 to configure a domain-specific, runtime state machine. The statemachine is implemented in a high-level language

(Objective-C, C, C++) for best performance. The high-level API is exposed through Lua to provide a user-friendly API to designers. The Lua scripts are run once to initialize the statemachine. (Battleforge, Spellforce)

Number 3 is already implemented by Kobold2D. Number 2 and/or 4 could be implemented in the future, provided that there's a common module that the scripts are supposed to control. For example, this could be a scriptable menu system so that menu screens can be designed and programmed entirely in Lua. This is **not** a promise for a feature, just something that *could* be done.

Number 4 is quite powerful, but new developers repeatedly find it confusing to write scripts that are executed exactly once (eg each time the level is restarted) because the scripting API implies a runtime nature that it doesn't have — it's only a setup/init script that's translated to a runtime state-machine.

Number 1 is not compelling — for one users can simply use the Corona SDK if they want to write everything in Lua, and secondly some of the drawbacks for Wax still apply: no debugging, no compile-time error checking, no Xcode support.

In addition, implementing Number 1 on top of Cocos2D means that there will have to be some amount of hacks, compromises and extension code to keep the Lua API clean and simple while making it work with the already existing Cocos2D API. It would make more sense to write an entirely new game engine with a Lua interface, and also make that cross-platform. But wait, that's what the Corona SDK is! You also get such a Lua interface with Cocos2D-X.

Kobold2D Audio FAQ

Which audio engine is better - CocosDenshion or ObjectAL?

That's subjective. For the most part.

CocosDenshion works on both iOS and Mac OS X. So if you do develop for both platforms you should use CocosDenshion.

If you are developing only for iOS, ObjectAL is a serious contender for two reasons: the API is very well designed, and it's well documented.

Audio Engine Comparison Chart

	CocosDenshion	ObjectAL
iOS	✅	✅
Mac OS X	✅	❌
"Simple" Audio Engine	✅	✅
Documentation	CocosDenshion Cookbook CocosDenshion FAQ	ObjectAL Reference Documentation

Kobold2D Physics FAQ

Which Physics Engine is the best?

Allow me ...

There's no "The Best"

There's only the physics engine that **better suits you**. Just like music.

	Box2D	Chipmunk
Language	C++	C
Documentation	Manual & API Reference	Manual & API Reference
Arithmetics	`vec = vec1 + vec2 - vec3;`	`vec = ccpSub(ccpAdd(vec1, vec2), vec3);`
Optimized for	Objects of varying sizes	Objects of similar sizes

Special Features	Bullets (aka continuous collision integration aka swept collisions) Operator overloading	-
Requirements	Must use .mm extension Objects should be sized between 0.1 to 10 units	-

There's also Objective Chipmunk, a commercial Objective-C wrapper for Chipmunk by the Chipmunk developer.

A free and popular Objective-C wrapper alternative is the Chipmunk SpaceManager (included in Kobold2D), but it is tied into Cocos2D and whenever the Cocos2D API changes, it requires the SpaceManager code to be updated accordingly. In the past it sometimes took weeks or months before the official SpaceManager code was updated. The SpaceManager is also not well documented, providing only an introduction and the API Reference.

There's no similarly popular Objective-C wrapper for Box2D. The CCBox2D project seems promising but like SpaceManager it's not a generic wrapper but depends on Cocos2D.

Which one is easiest to learn?

Again, this depends on you. 50% of developers who swear that Box2D is easier to learn than Chipmunk fail to understand why the other 50% find Chipmunk easier to learn than Box2D.

The % are made up, but you get the point. 

Things to consider before deciding on a physics engine

Some people would think the choice should mainly be based on performance. Wrong. Usability, familiarity with the programming language and documentation are far more important in choosing any library almost always. Even more so if you have little to no experience with physics engines.

There are some special cases where Chipmunk may be faster, other cases where Box2D may be faster. If you know what cases that might be you should perform some simple tests and measure the performance. Note that Chipmunk SpaceManager will always be slower than pure Chipmunk, but it makes up for being easier on you if you already know and like Objective-C.

If you know what kind of joints you are likely going to need, check the documentation for each physic engine. Chipmunk may have some joint types that Box2D doesn't have, and vice versa. This could easily be a killer argument for or against a certain physics engine.

If you are going to have very fast moving physics objects, eg "Bullets", consider using Box2D as it can do swept collisions aka continuous collision integration to prevent fast-moving objects from deeply penetrating or even tunneling through other objects.

If you find one physics engine more "accurate" or "stable" than the other, eg. stacking objects wiggle more or less, know that some of these issues can be resolved by changing the number of iterations the physics engine performs. This also affects performance. It is not unusual for iOS games to reduce the number of iterations to 1 or 2 for performance reasons. You might want to try that and see which physics engine requires how many iterations to simulate your game with the needed accuracy before introducing issues like "restless" objects.

If you prefer object-oriented development, have a history with C++ or prefer speaking variable names, you should choose Box2D. But in Kobold2D projects you will have to change the .m file extension to .mm and stick with that to compile the code as Objective-C++ code. Failure to do so even once will greet you with dozens, if not hundreds of compile errors. Don't Panic!

If you are a purist and prefer C and don't mind dealing with variable names like `e`, `f` and `m` then try Chipmunk or Chipmunk SpaceManager. Chipmunk code is also not as verbose as Box2D code due to its naming scheme, but then again it doesn't provide overloaded operators so all arithmetics are done more verbose and less intuitive via function calls: `ccpVect vec = ccpSub(ccpAdd(vec1, vec2), vec3)`.

Kobold2D Cocos3D FAQ

Is there a Mac OS version of Cocos3D?

No.

It's listed on the development roadmap, but very far down.

Kobold2D Release Notes

Read the release notes after upgrading Kobold2D to learn about changes that may affect you.

Kobold2D v1.0 Preview 3 Release Notes

The Most Important Changes & Additions

- Added Isometric Tilemap project template.
- Added Sprite Performance project template. Demonstrates the use and performance of CCSpriteBatchNode and PVR.CZZ texture atlas vs individual sprites from PNG files.
- Added Mac version of Doodle Drop.
- Added cocos2d-iphone-extensions develop branch (to support TMXGenerator)

Minor Improvements & Bug Fixes

- Fixed missing header search paths to cocos2d-iphone-extensions
- Fixed release build failure of Pinball game template
- Added missing "usesPremultipliedAlpha" call to Parallax Game template

Known Issues

- Incompatible with iOS SDK 5 **Preview**. See: Does Kobold2D work with the latest beta SDK?
- Modified library code (eg. cocos2d-iphone) may not be linked into the App as it should be. This is an Xcode bug, see this thread for causes and solutions.

All Release Notes

Kobold2D v1.0 Preview 1 Release Notes

Important Changes

- Initial Release, everything is new.

Known Issues

- Incompatible with iOS SDK 5 **Preview**. See: Does Kobold2D work with the latest beta SDK?
- Project names with space(s) in them (eg "My Project") will report errors like

```
Prefix-iOS.pch:27:20: error: cocos2d.h: No such file or directory
```

- Not all template projects included yet (eg. Pinball game, Tilemap demos, UIKit Integration).
- Project Starter tool does not yet support adding projects to a custom workspace.
- Project Upgrade tool not yet included (not needed until preview 2).

Kobold2D v1.0 Preview 2 Release Notes

Important Changes and New Features

- Added Kobold2D Project Upgrader tool.
- Kobold2D Project Starter tool now allows you to select and create custom workspaces.
- Added two template projects: Orthogonal Tilemap & Pinball Game
- updated cocos3d to v0.6.1
- fixed failing builds due to spaces in project name or username

Bug Fixes and Minor Improvements

- added help button to tools

- Fixed: spaces in project names or path to Kobold2D (if moved from its default location) should no longer break the build with "missing file or directory" errors (pertaining to library headers)
- Fixed: Doodle Drop horizontal motion is inverted when playing in portrait upside down mode.
- Fixed: crash in FontLabel trying to load a .ttf font.
- Parallax side scroller doesn't appear correct on the iPad. Added note that background images weren't designed for iPad dimensions.
- Physics template projects not working correctly on Retina devices. Disabled Retina mode since these projects do not use HD images.

Known Issues

- Incompatible with iOS SDK 5 **Preview**. See: Does Kobold2D work with the latest beta SDK?
- Some template projects still missing (eg. UIKit demos, Iso Tilemap).
- Modified library code (eg. cocos2d-iphone) may not be linked into the App as it should be. This is an Xcode bug, see this thread for causes and solutions.

Kobold2D v1.0 Preview 3 Release Notes

The Most Important Changes & Additions

- Added Isometric Tilemap project template.
- Added Sprite Performance project template. Demonstrates the use and performance of CCSpriteBatchNode and PVR.CZZ texture atlas vs individual sprites from PNG files.
- Added Mac version of Doodle Drop.
- Added cocos2d-iphone-extensions develop branch (to support TMXGenerator)

Minor Improvements & Bug Fixes

- Fixed missing header search paths to cocos2d-iphone-extensions
- Fixed release build failure of Pinball game template
- Added missing "usesPremultipliedAlpha" call to Parallax Game template

Known Issues

- Incompatible with iOS SDK 5 **Preview**. See: Does Kobold2D work with the latest beta SDK?
- Modified library code (eg. cocos2d-iphone) may not be linked into the App as it should be. This is an Xcode bug, see this thread for causes and solutions.

Kobold2D v1.0 Preview 4 Release Notes

This Kobold2D version is not available yet!
This page describes changes to Kobold2D which have been implemented, but the Kobold2D version it refers to has not been released yet.

The Most Important Changes & Additions

- Added KKInput class to poll Mac keyboard & mouse states, and on iOS tracks accelerometer, gyroscope, deviceMotion ...
- Added User Input template project to illustrate how to use KKInput
- Added Empty (Minimal) Project
- Fixed archive builds not being able to create .IPA in Organizer.
- Fixed .c files causing compile errors due to mistake in prefix pch headers.

Minor Improvements & Bug Fixes

- Fixed CocosDenshion CDXPropertyModifierAction not available in iOS builds
- Fixed Box2D headers in prefix headers now enclosed in `#ifdef __cplusplus`
- Fixed static library changes not being linked with app, unless build is "cleaned"
- Fixed Mac window: disable resize, remove "status bar" area at bottom, set AutoScale=NO as default, changed config.lua windowFrame to be the size of the GL view (takes title bar height into account)
- Mac info.plist: set KKApplicationMac as PrincipalClass to intercept Command key events
- LOG_EXPR(variable) now also works in Mac builds
- removed "strict selector matching" warning in template projects
- changed several template projects' first class to derive from CCLayer instead of CCScene

Known Issues

- Incompatible with iOS SDK 5 **Preview**. See: Does Kobold2D work with the latest beta SDK?

Kobold2D Contributor's Guide

Contributing

How to become a Kobold2D Contributor
Kobold2D Folder Structure Explained

Step by Step

Creating a new Project Template for Kobold2D
Adding a new Xcode File Template to Kobold2D
Adding a new Static Library to Kobold2D

Guidelines

Managing Kobold2D Build Settings
Guidelines for Kobold2D Development
Guidelines for Commercial Add-On Products

Kobold2D Build Status

The Build Status will be updated within minutes after a source code modification and indicate whether the code was built successfully or not.

Build Status not yet implemented